Computational Neuroanatomy of the Central Complex of *Drosophila melanogaster*

Mark Longair

PhD by Research

The University of Edinburgh

2009

To Jenny

Contents

Abbreviations

♂	Male
♀	Female
☿	Virgin Female
CS	Canton S: a wildtype fly strain
CyO	Curly of Oster: a 2nd chromosome balancer
DAG	Directed Acyclic Graph
eb	Ellipsoid Body
EM	Electron Microscopy
fb	Fan-shaped Body
HFS	Horizontal Fibre System
HU	Hydroxyurea
MASC	MAGUK Associated Signalling Complex
MI	Mutual Information
MIP	Maximum Intensity Projection
NRC	NMDA Receptor Complex
OPT	Optical Projection Tomography
OrR	Oregon R: a wildtype fly strain
pb	Protocerebral bridge
PF	Paraformaldehyde
PBS	Phosphate Buffered Solution
PBT	A solution of 0.3% Triton X-100 in PBS
PMT	Photomultiplier Tube
sog	Sub-Œsophageal Ganglion

Abstract

In many different insect species the highly conserved neuropil regions known as the central complex or central body complex have been shown to be important in behaviours such as locomotion, visual memory and courtship conditioning. The aim of this project is to generate accurate quantitative neuroanatomy of the central complex in the fruit fly *Drosophila melanogaster*. Much of the authoritative neuroanatomy of the fruit fly from past literature has been derived using Golgi stains, and in important cases these data are available only from 2D camera lucida drawings of the neurons and linguistic descriptions of connectivity. These cannot easily be mapped onto 3D template brains or compared directly to our own data. Using GAL4 driver and reporter constructs, some of the findings within these studies could be visualized using immunohistochemistry and confocal microscopy. A range of GAL4 driver lines were selected that particularly had prominent expression in the fan-shaped body. Images of brains from these lines were archived using a web-based 3D image stack archive developed for the sharing and backup of large confocal stacks. This is also the platform which we use to publish the data, so that other researchers can reuse this catalogue and compare their results directly. Each brain was annotated using desktop-based tools for labelling neuropil regions, locating landmarks in image stacks and tracing fine neuronal processes both manually and automatically. The development of the tracing and landmark annotation tools is described, and all of the tools used in this work are available as free software. In order to compare and aggregate these data, which are from many different brains, it is necessary to register each image stack onto some standard template brain. Although this is a well-studied problem in medical imaging, these high resolution scans of the central fly brain are unusual in a number of respects. The relative effectiveness of various methods currently available were tested on this data set. The best registrations were produced by a method that generates free-form deformations based on B-splines (the Computational Morphometry Toolkit), but for much faster registrations, the thin plate spline method based on manual landmarks may be sufficient. The annotated and registered data allows us to produce central complex template images and also files that accurately represent the possible central complex connectivity apparent in these images. One interesting result to arise from these efforts was evidence for a possible connection between the inferior region of the fan-shaped body and the beta lobe of the mushroom body which had previously been missed in these GAL4 lines. In addition, we can identify several connections which appear to be similar to those described in [Hanesch et al., 1989], the canonical paper on the architecture of the *Drosophila melanogaster* central complex, and describe for the first time their variation statistically. This registered data was also used to suggest a method for classifying layers of expression within the fan-shaped body.

1 Introduction

The ultimate motivation for the work presented here is very ambitious: namely to gain a better understanding of learning and memory in the human brain by modelling those functions in the fruit fly *Drosophila melanogaster*. The purpose of this introduction is to link that aim with the content of this thesis and then set out the approach that I have taken, which is centred around the application of computer science techniques to biological problems.

1.1 Motivation

Drosophila melanogaster's use as a model organism dates back to the pioneering work of Thomas Hunt Morgan early in the 20th century, who chose it as a subject for practical explorations of inheritance and mutation, following on from the work of Mendel and Darwin. The reasons suggested for this choice were largely pragmatic ones: fruit flies are cheap to maintain, have a short generation time (10 days), are easy to breed and, compared to mammalian model organisms, do not take up much space in a laboratory [St Johnston, 2002, Weiner, 2000]. As a result of being used by so many early researchers and their spirit of cooperation, the genetic toolkit for *Drosophila melanogaster* which has been developed over the last 100 years is quite extraordinary. This progress included the major landmark of the publication of a largely complete sequence of the genome in 2000 [Adams et al., 2000, Ashburner, 2006]. A particularly notable genetic technique used widely in *Drosophila* research is the GAL4/UAS driver system developed by Brand and Perrimon, which allows expression of arbitrary transgenes only in the cells in which a particular enhancer is expressed [Brand and Perrimon, 1993]. The upshot of this is that whereas testing hypotheses regarding neuronal circuits in mammals usually requires surgical intervention, in the fruit fly it is often possible to do analogous operations genetically, such as, for example, suppressing neurons selectively and reversibly with the temperature sensitive *shibire* mutation [Kitamoto, 2001]. Another remarkable recent advance is to be able to selectively activate neurons with LASER light [Lima and Miesenböck, 2005]. Further examples of the power of genetic manipulation of the *Drosophila* brain can be found in [Vosshall, 2007].

The brain of *Drosophila melanogaster* is vastly less complex than the human brain. For example, typical estimates of the number of neurons in the central nervous system (CNS) of *Drosophila* are around 10^5 compared to the order of 10^{12} for *Homo sapiens* [Bear et al., 2006, Armstrong and van

Hemert, 2009]. Despite this, the fruit fly displays a remarkably sophisticated range of behaviours. In particular, they are capable of a variety of different types of learning and memory, detectable via assays such as olfactory avoidance, courtship conditioning and visual learning paradigms. Different patterns of training in olfactory avoidance tests have been demonstrated to produce memories with distinct persistence and genetic dependencies, from short term memory and middle term memory (both lasting a matter of hours) to anæsthesia-resistant memory and long term memory (which lasts days) [Isabel et al., 2004]. Long term memory is distinct from the other types of memory in that it is dependent upon protein synthesis. Learning and memory in *Drosophila* is reviewed comprehensively in [Dubnau and Tully, 1998] and [Keene and Waddell, 2007]. There is sufficient homology between the human and fruit fly genomes that it is possible to create *Drosophila* models of human neurodegenerative diseases, including those with memory symptoms such as Alzheimer's disease [Jeibmann and Paulus, 2009]. Among the many differences between human and *Drosophila* neural architecture is that neurons in the former are typically unipolar, with cell bodies around the outside of the brain, as opposed to heteropolar, but there are many reasons to believe that they are nonetheless functionally homologous [Sánchez-Soriano et al., 2005].

One approach we have taken to finding interesting genes and brain regions linked to learning and memory in the fruit fly is to look for homology between the mammalian NRC/MASC[1] and the *Drosophila melanogaster* proteome. The NRC/MASC is made up of 186 proteins, discovered via immunoprecipitation, which make up part of the post-synaptic machinery of mammalian neurons [Husi et al., 2000, Pocklington et al., 2006, Emes et al., 2008]. In some early work with Dr Douglas Armstrong, I screened a large set of commercial antibodies to proteins in the mammalian NRC/MASC for those whose antigens had a strong similarity to proteins in *Drosophila* [Longair, 2004]. Notably, we discovered that three antibodies suggested expression of the genes *rl* (orthologue of mouse ERK-2), *Rop* (orthologue of mouse STXB1) and *Ph1* (orthologue of mouse MAPKK) in a structure in the fly brain known as the fan-shaped body. Although we did not go on to do Western blots to test for the specificity of these antibodies it suggested a further investigation of this structure as having a possible role in learning and memory. This structure is one of several that have been implicated as having some effect on learning and memory in the fruit fly, and these are briefly surveyed in the next section.

[1]NRC stands for the NMDA Receptor Complex, also known as MASC, the MAGUK Associated Signalling Complex. MAGUKs are Membrane-Associated Guanylate Kinases, a class of proteins with a PDZ, SH3 and GUK domain at the C-terminal end.

1.2 *Drosophila* Neuroanatomy and Learning and Memory

The region of the fruit fly brain that has been most studied with regard to learning and memory are the mushroom bodies. These bilaterally symmetric paired neuropil structures (see Figure 1) are highly conserved across insects and in *Drosophila melanogaster* have been shown to be necessary in order to learn to discriminate between odours [Heisenberg et al., 1985, de Belle and Heisenberg, 1994]. An excellent review of the role of the mushroom bodies, focussing on *Drosophila melanogaster* but also discussing other insect species, can be found in [Heisenberg, 2003]. Because of the central role of the mushroom bodies in olfactory learning, it had been hypothesized that this structure may be a general substrate for learning in the fruit fly, but Wolf et al. [1998] demonstrated that there are several learning assays, covering both operant and classical conditioning, that are unaffected by complete ablation of the mushroom bodies with hydroxyurea (HU). This included a variety of visual learning assays and courtship conditioning.

The other neuropil regions that have most significantly been implicated in learning and memory in *Drosophila* are a collection of four structures known as the central complex, consisting of areas known as the fan-shaped body, the ellipsoid body, the protocerebral bridge and the noduli.[2] Mutant lines with disruptions to the central complex were shown to have poor olfactory learning in [Heisenberg et al., 1985].

More recently a number of papers have been published that link layers of the fan-shaped body to visual memory. The first of these, [Liu et al., 2006], found layers of the fan-shaped body (dubbed F1 and F5 in that paper) that were necessary for learning and remembering two visual features, namely contour orientation (F1) and elevation (F5). Later, [Wang et al., 2008] demonstrated that expression of a functional *for* transgene in either those same layers of the fan-shaped body or the R2 and R4m neurons of the ellipsoid body was sufficient to rescue various visual memory defects in flies with a particular mutation in the *for* gene.

The proposed functions of the central complex are not limited to learning and memory. [Strausfeld, 1999] makes a strong case that the central complex is involved in the higher control of walking in insects. [Martin et al., 1999] and [Strauss, 2002] showed that disruptions to the central complex

[2]The ventral bodies (sometimes known as the lateral accessory lobes), which it is suggested may collect output from the central complex [Hanesch et al., 1989], are sometimes included in the regions referred to as the "central complex", e.g. in [Strausfeld, 1976]. In this thesis, however, I will explicitly mention the ventral bodies if they are grouped with the other neuropil regions.

A

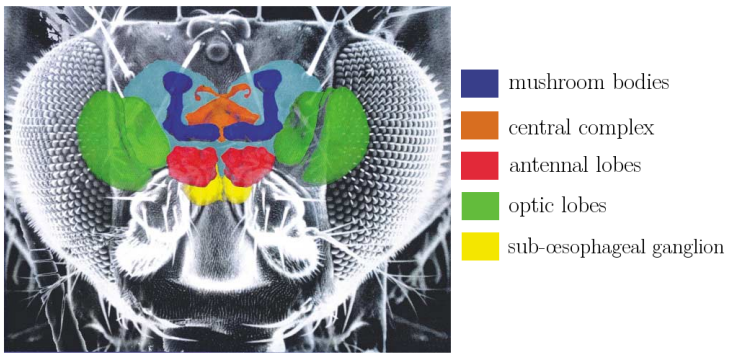
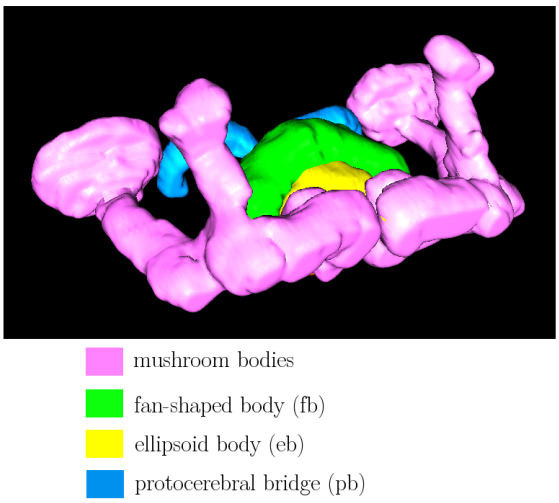


Image reproduced from [Heisenberg, 2003]

B



C

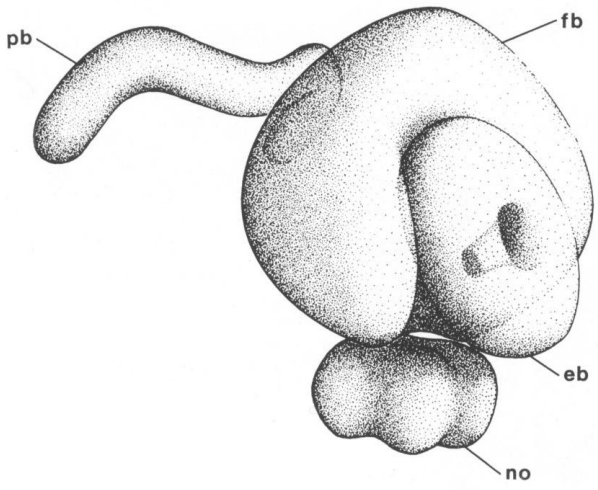


Image reproduced from [Hanesch et al., 1989]

Figure 1 – Important neuropil regions of the brain of *Drosophila melanogaster*. A: select neuropils shown in the context of the fly's head capsule, taken from [Heisenberg, 2003]; B: the central complex surrounded by the mushroom bodies, rendered from my own data; C: a schematic drawing of the central complex elements taken from [Hanesch et al., 1989].

disrupt the maintenance of locomotor activity, although its initiation is unaffected[3]. The central complex has also been linked to ethanol tolerance [Scholz et al., 2000] and courtship conditioning and behaviour [Ilius et al., 1994, Sakai and Kitamoto, 2006]. In the desert locust, *Schistocerca gregaria*, neurons arborizing in the upper and lower central body (structures homologous to the fan-shaped body and ellipsoid body respectively) have been shown to be sensitive to polarized light, suggesting that in that organism it may be linked to orientation with respect to the sun and sky [Vitzthum et al., 2002].

[Zars et al., 2000] presents perhaps the most surprising links between learning and memory in the fruit fly and its neuroanatomy. By driving expression of *rutabaga* in a null mutant background under control of various different GAL4 driver lines, the authors found several lines that rescued performance in a spatial learning paradigm. The brain structures that showed expression in all of these lines were the antennal lobes, the median bundle and ventral ganglion. However, there was expression in either the fan-shaped body or ellipsoid body in every rescuing line, and the later result in [Wang et al., 2008], which shows that *for*-dependent learning may be rescued by expression in *either* of these structures, perhaps suggests that it may not be safe to exclude these regions based on this criteria.

It is worth noting that many of the behavioural results linked to the central complex have to be interpreted with care because of the following points:

- There are many studies based on the Würzburg central complex mutants (Figure 2) such as [Heisenberg et al., 1985], [Strauss and Heisenberg, 1993], [Martin et al., 1999], [Sakai and Kitamoto, 2006], [Scholz et al., 2000], etc. These mutants have many complex behavioural phenotypes, described in detail in, for example, [Strauss et al., 1992] for *no-bridge* and [Ilius et al., 1994] for *ellipsoid body open*. [Carhan et al., 2005] describes the brain structure phenotypes of one of the *central brain deranged* strains as being highly variable. Since the phenotypes in these mutants vary significantly from fly to fly, so they must be dissected and examined afterwards to discover the degree of disruption to the central complex. In addition, because there are so many behavioural differences between these mutants and wild-type flies, designing experiments that are insensitive to these confounding factors may be difficult.

[3] [Heisenberg, 2003] hypothesizes that the *initiation* of locomotion, or decisions more generally, may be mushroom body dependent.

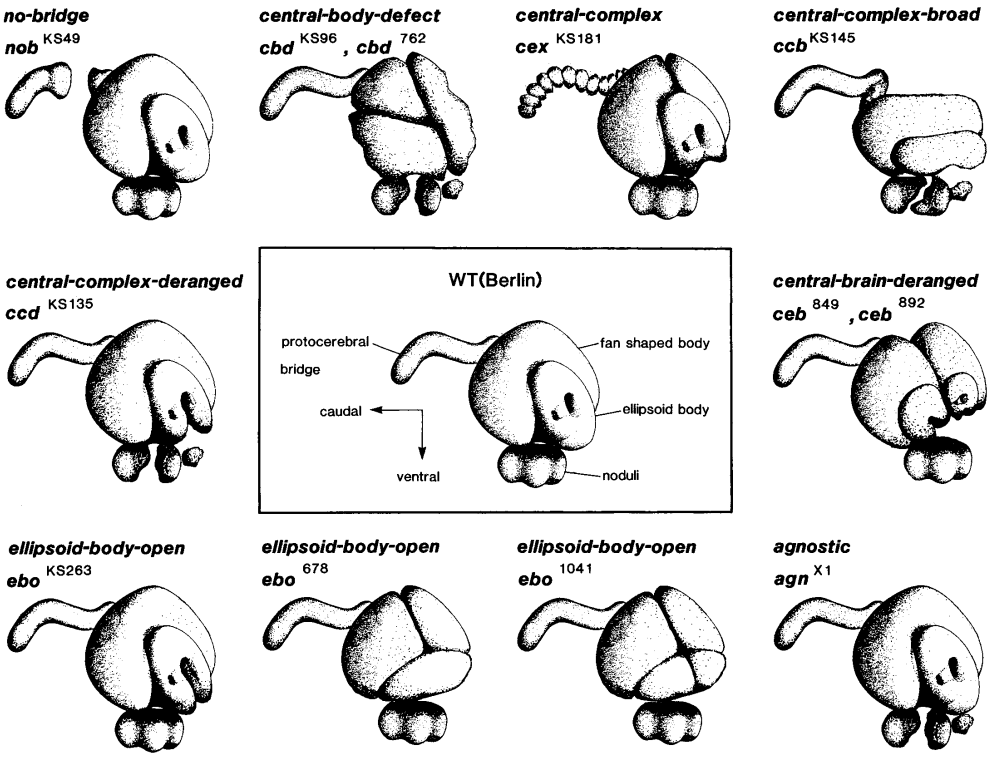


Figure 2 – An image of the central complex mutants used in [Strauss and Heisenberg, 1993], reproduced from that paper.

- [Liu et al., 2006] (and the subsequent paper [Wang et al., 2008]) use GAL4 lines that have prominent expression in particular areas of the fan-shaped body, replicating the results for multiple lines whose expression overlaps in the same layers but has apparently very low co-localization in the optic lobes. There is no analysis presented in these papers of the co-localization between these lines after registration throughout the whole brain, which leaves some doubt about whether the fan-shaped body layers in question are in fact the only significant commonality between the expression patterns. A similar difficulty arises with interpreting the results of [Zars et al., 2000]. Techniques for testing the significance of colocalization can be found in [Costes et al., 2004].

1.3 Central Complex Neuroanatomy

The neuroanatomy of the central complex has been extensively studied in several insect species, most notably *Musca* [Strausfeld, 1976] and *Schistocera gregaria* [Williams, 1975, Heinze and Homberg, 2008]. In this brief review, however, I will largely concentrate on work specifically done in *Drosophila melanogaster*. The distinctively shaped central complex neuropils in the fruit fly, which are surrounded by the lobes of the mushroom bodies (Figure 1), have many remarkable features. These highly-interconnected, bilaterally-symmetric regions span the midline of the brain. Some of their interesting structural features are clear even from high quality confocal scans of neuropil markers, such as:

- The near-horizontal layers of the fan-shaped body, perpendicular to the inferior-superior axis;
- The central "segments" (or "staves") of the fan-shaped body running in the inferior-superior direction;
- The glomerular structure of the protocerebral bridge;
- The rotational symmetry of the ellipsoid body.

In averaged images from many brains, there is also a clear distinction between the synaptic density in the inner and outer rings of the ellipsoid body.

However, for a more detailed dissection of the neuroanatomy of these regions, different brain preparations and more specific neural markers are required. The classic paper describing the remarkable, regular structure of neurons passing through the central complex, and in particular the

lattice of connectivity within the fan-shaped body, is [Hanesch et al., 1989]. This study was based on over 1000 Golgi preparations of fly brains and presents camera lucida drawings of single neurons. In addition to the grosser structural features of the central complex mentioned above, [Hanesch et al., 1989] describes a division of the fan-shaped body in a frontal-occipital direction into four shells. The paper uses the appearance of the arborizations branching off from the neurons to suggest neuronal polarity: "spiny" terminals are assumed to be dendritic (i.e. input regions) and "blebbed"[4] terminals are assumed to be axonal (i.e. output regions). Following [Hanesch et al., 1989]'s classification, I will summarize the known neuron types in sections 1.3.1 and 1.3.2 below. Essentially, the large field neurons connect a complete stratum of one of the elements of the central complex (e.g. a layer of the fan-shaped body, the complete protocerebral bridge, etc.) to some other region of the brain while the small field neurons connect small arbors between (or within) elements of the central complex, the ventral bodies and lateral triangles.

1.3.1 Small Field Neurons

There are many small field neurons which have descriptive names such as "pb-eb-no",[5] and which are summarized in Table 1 of [Hanesch et al., 1989], so I will not list them exhaustively here. (These classes include connections from the fan-shaped body to every other element of the central complex, for example.) However, there are several classes of small field neurons with less obvious nomenclature, including:

Pontine Neurons: The paper describes pontine neurons as those that connect small arbors within a structure, but the only examples given are those with two arbors in the fan-shaped body. There are four sub-types described, whose two arbors are either:

1. On either side of the midline in a single layer, skipping three segments;
2. In two adjacent segments in a single layer;
3. In two adjacent layers in a single segment;
4. In the inferior and superior layers of a single segment.

[4]This term, derived from "blebs", a word for mushroom-like protrusions from cells, suggests a bouton-like appearance.

[5]i.e. a neuron with arborizations in the protocerebral bridge, ellipsoid body and the noduli.

[Young and Armstrong, 2009, in preparation] suggests the nomenclature P1 to P4 for these subtypes.

Vertical Fiber System:[6] This describes a set of neurons with one arbor in one of the 16 glomeruli of the protocerebral bridge, one in a topographically corresponding segment of the fan-shaped body and a final arborization in one of the contralateral noduli.

Horizontal Fiber System: The horizontal fiber system similarly has a network of connections from single glomeruli of the protocerebral bridge to the contralateral ventral body, except for the glomeruli at each tip of the protocerebral bridge, which are connected to the ipsilateral ventral body. Each of these connections also arborizes in a predictable segment of the fan-shaped body on the way, as shown in Figure 6b of [Hanesch et al., 1989].

1.3.2 Large Field Neurons

[Hanesch et al., 1989] reports discovering over 200 distinct large field neurons, which are only broadly classified. The most significant class of these neurons for the work in this thesis are those whose central complex arborizations lie in the fan-shaped body, known as the fan-shaped neurons. Each of these completely fills one or more adjacent layers of the fan-shaped body. There are two readily identifiable types of fan-shaped neuron described in [Hanesch et al., 1989], known as Fm and Fl neurons:

Fm (medial): In this sub-class of fan-shaped neuron the process leading to the fan-shaped body arbor passes through the ellipsoid body canal.

Fl (lateral): In this sub-class of fan-shaped neuron the process leading to the fan-shaped body arbor instead approaches the fan-shaped body laterally and does not pass through the ellipsoid body.

The paper describes numbered types of Fm neurons (Fm1, Fm2, Fm3) depending on the position of the cell bodies, and suggests that there is enough variation in the features of all types of fan-shaped neurons that there are likely to be further identifiable subtypes.

[6] In order to match the spelling in the paper, I have preserved the U.S. English spelling of "fibre" in this phrase.

There are a few types of large field neuron that fill the protocerebral bridge. Otherwise, the large field neurons that are most frequently seen in the central complex are the ring neurons, which arborize in the ellipsoid body. Excluding the rarely found ExR1 and RxR2 type neurons, the most prominent large field neurons have been named R1 to R4, with the later paper [Renn et al., 1999] subdividing R4 further into R4m and R4d.

R1, R2, R3, R4m, R4d: The R neurons all have in common a cell body in the frontal side of the brain by the antennal lobes and an apparently dendritic region in the lateral triangles. The R1 to R3 neurons enter the ellipsoid body medially while the R4 neurons enter at the lateral edge. The R4 neurons have terminals around the greatest circumference of the ellipsoid body, slightly biased towards the frontal side, with the R4d type having a thinner ring more distally than the R4m type. The R1 type of neurons have terminals in the inner ring of the ellipsoid body, running from the frontal side through the occipital side, tapering off at that edge. The R2 type's terminals fill a similar region to the R4 neurons. The R3 neurons fill the frontal half of the ellipsoid body.

The description above provides an overview of the neuron types found in the central complex. The hypothesis put forward in [Hanesch et al., 1989] is that the large field neurons provide input from various parts of the brain, the complex network of small field neurons provides processing, and the output from the system is via the ventral bodies. However, the complexity of the interconnections allows for many other possible interpretations.

1.3.3 The Fan-Shaped Body and the Mushroom Body

An interesting outstanding question about *Drosophila* neuroanatomy is whether there might be a direct connection between the fan-shaped body and the mushroom bodies. Such a connection has never been described in the *Drosophila* literature, although apparently it exists between the homologous structures in *Formica*.[7] If such a connection were to exist, it would have a significant implication for models of learning and memory: it would raise the possibility that one of these two regions might be a common substrate for learning and memory in the fruit fly. It had been suggested in our group that such a link may be present in a particular GAL4 driver line, known as c061. This was a subject of some debate, however, and proved to be difficult to resolve by examining confocal

[7] I am grateful to Dr Joanna Young for pointing out that this is described in [Goll, 1967], although that paper is currently inaccessible to me due to not having an English translation.

stacks in a slice-by-slice manner. Testing this hypothesis became one of the motivations for the later work in this thesis, part of which enables us to precisely express connectivity hypotheses with computational annotations.

1.4 Computational Neuroanatomy

The amazing achievement of [Hanesch et al., 1989] and other such studies notwithstanding, there are some significant problems with the publication of connectivity data in the form of camera lucida drawings and linguistic descriptions in classical neuroanatomical terms. At the time of that paper, of course, there was no alternative, but since then there have been advances in image acquisition, computational power and image processing that make new ways of dealing with such data possible: I think that the term "computational neuroanatomy" captures these possibilities well.

From early in the computer revolution, the potential for using computers to aid neuroanatomists in recording, analysing and presenting data was clear. This early history of computational neuroanatomy is well described in [Capowski, 1989], which looked at the practical use of computers for tracing neurons directly from the microscope, reconstructing neurons from serial EM photographs and basic 3D visualizations. Notably, the book also discusses the problems with fully automatic neuron tracing, which is still a significant challenge today.

Nowadays, the vast majority of biological image data used by scientists is acquired by computer in the first place, but there are still significant obstacles to using this data computationally. Regrettably, a large part of this problem is with the way that image data are typically published by journals. With some notable exceptions (e.g. [Jefferis et al., 2007]) it is still very rare for authors to publish image stacks online with the publication of their paper. This reduces the original data to a form which has the perennial problem of paper publication: the depth of the images is lost, which makes it impossible to reconstruct a 3D representation of any structure of interest. One of the most common effects of this is that it makes it very difficult to assess whether a structure in one's own data is likely to be the same as one in a publication. Even in small and easily recognizable regions of the *Drosophila* brain such as the central complex this can be a problem, let alone the large regions of the protocerebrum that do not have such helpful landmarks. For example, it is difficult from the descriptions of the paths of large field neurons in [Hanesch et al., 1989] to tell where their cell bodies typically lie or how to distinguish layers of the fan-shaped body. As a more recent example, it is

awkward to decide whether some of the neurons in [Li et al., 2009] are the same as those acquired in our own images.

The question of how to publish such data online easily is one that will be addressed further in Chapter 3. In other respects, techniques that might be regarded as computational neuroanatomy are already a ubiquitous part of many researchers' jobs. However, it is perhaps worth articulating how remarkable the possibilities for image processing and analysis are with computational tools. To take some examples:

- Using image registration, it is possible to directly compare image data from different fly brains or aggregate data from multiple subjects to generate atlases [Brandt et al., 2005] or precisely map neuronal projection patterns [Jefferis et al., 2007].
- High-speed 3D visualization gives the user a much more intuitive and unbiased view of data.
- Co-localization of proteins can be assessed using statistically sound methods, e.g. [Costes et al., 2004].
- Expression patterns can be automatically classified according to their location in the brain.
- Neurons can be traced, reconstructed, and coloured according to the expression of proteins through their structure [Evers et al., 2005].
- The computer can remove some of the tedium from hand annotation of data for analysis, such as cell-counting.
- One can use label maps or electronic atlases to assist researchers in learning and consistently identifying areas of the brain.[8]
- It is possible to produce community resources such as large databases of expression patterns (e.g. the Allen Mouse Brain Atlas,[9] or the Edinburgh Mouse Atlas Project[10])

[8] For example: `http://fruitfly.inf.ed.ac.uk/brain/`

[9] `http://mouse.brain-map.org/`

[10] `http://genex.hgu.mrc.ac.uk/`

1.5 Goals of this Project

The aim of this work is to generate high quality neural connectivity information for the fan-shaped body using GAL4 driver lines and confocal microscopy. This data will be published online with tracing annotations and the original source data. Ultimately, we hope that variations on this work will enable studies of the complexity of [Hanesch et al., 1989] to be carried out more easily, and to generate high quality data sets that can be easily reused by other researchers. A still more distant goal, which is most likely many years away, is to be able to generate an atlas of connectivity data of sufficient quality in order to simulate the fly brain (or defined circuits of it) at a neural network level [Armstrong and van Hemert, 2009].

There are also two specific neuroanatomy questions which I attempt to address:

- Can the image data support the hypothesis that the line c061 shows a direct connection between the fan-shaped body and mushroom body? The obstacles to testing this were essentially very practical ones: (a) we needed image data from more flies in order to be sure that the hypothesized connection was not an artefact of imaging or due to a mutation and (b) we required software tools even in order to annotate precisely where the posited connection lies.
- Can we find evidence from our image data to either support or contradict the suggestion in [Hanesch et al., 1989] that there are 6 layers in the fan-shaped body? The collection of GAL4 lines maintained by the Armstrong group [Yang et al., 1995] contains many lines that have clear layered expression in the fan-shaped body, so this is an ideal question for our group to approach using computational neuroanatomy.

The remaining sections of this introduction discuss methodological and pragmatic concerns relevant to the work described in the following chapters.

1.6 Methodology

1.6.1 Free Software

There exists a huge market in scientific software aimed at biologists for the visualization and analysis of data, pertinent examples of which for this thesis are Amira and Neurolucida. While these are

high quality packages that work well for many researchers, they are not appropriate for work such as this, where the development and evaluation of algorithms is as much part of the investigation as the eventual results. In this thesis I am working towards a complete pipeline of "free software" or "open source" tools for the study of neuroanatomy. In the phrase "free software", "free" is intended to have the meaning of "at liberty" instead of "gratis"; the Free Software Foundation's criteria for such software are:

> "The freedom to run the program, for any purpose (freedom 0).
>
> The freedom to study how the program works, and change it to make it do what you wish (freedom 1). Access to the source code is a precondition for this.
>
> The freedom to redistribute copies so you can help your neighbor (freedom 2).
>
> The freedom to improve the program, and release your improvements (and modified versions in general) to the public, so that the whole community benefits (freedom 3). Access to the source code is a precondition for this."[11]

In addition to the monetary saving in not having to purchase expensive commercial software, this approach has the following advantages for this work:

- Where the source code is not freely available for a product, it can be extremely difficult to get bugs in the software fixed, whereas with free software it is always possible to create your own patch to correct the problem. This is far from a theoretical concern: the Virtual Insect Brain project moved from the platform of Amira to the public domain software ImageJ because they found that certain bugs were never fixed.[12] In addition, when companies go out of business, the source code for products may be lost, or sold on to other companies who have no interest in developing it further.

- While some companies do have an admirable record with publishing papers on their algorithms, this is far from universal. In addition, as anyone who has tried to implement an algorithm based solely on a short description in a paper will know, it is often very difficult to reproduce the method without more details about the implementation. In the alternative free software world, every detail of an implementation is available in the source code of the program.

[11] From `http://www.gnu.org/philosophy/free-sw.html` on 2009-08-27.

[12] Private communication from Dr Johannes Schindelin.

- There is a dedicated and enthusiastic community of software developers and researchers who work on free software for scientific applications. This is an invaluable resource: this open community helps one to exchange ideas, share source code and develop and evaluate software more quickly than would otherwise be possible.

- By releasing software early, and with source code, interested developers can help to track down bugs and problems at a deeper level than would otherwise be possible. This may mean the difference between a bug report with a description of the circumstances in which something goes wrong and a report with a patch that fixes the bug in one's source code.

As well as these pragmatic considerations, there is a more philosophical one: when scientific research is published, it should be in a form such that it is possible to reproduce and adapt the methods used. In publications whose results depend critically on software, this means publishing the source code under a free software licence.

1.6.2 The Fiji Project

At an early stage in this work I was encouraged by Dr Johannes Schindelin and Dr Arnim Jenett to consider using ImageJ as a platform for my work. ImageJ is a public domain Java-based image processing tool developed by Wayne Rasband at NIH, which has a large and active community of users and developers. This turned out to be a very fruitful suggestion, since developing software as ImageJ plugins made it easy to support multiple operating systems and enabled many useful collaborations. Such collaborations led to the creation of the Fiji project, started by Dr Johannes Schindelin and Dr Albert Cardona, which is a packaging of ImageJA[13] with a large selection of useful plugins for segmentation, registration, reconstruction, analysis, and so on. The completed plugins described in this thesis are all available as part of Fiji.

Fiji's significance is that it provides a powerful platform of reusable free software components for biological image analysis, which greatly reduces the amount of code that one needs to write to achieve complex tasks. To take two examples:

- The scripting language support allows interaction with existing plugins in high level languages, such as Python and Ruby.

[13] ImageJA is a fork of ImageJ which closely tracks the latest versions of ImageJ.

- 3D visualization is easy since Fiji includes Benjamin Schmid's ImageJ 3D Viewer. This made adding 3D rendering in my Simple Neurite Tracer plugin relatively easy.

More information about Fiji is available at the project website: `http://pacific.mpi-cbg.de/`

1.7 Stylistic Conventions and Nomenclature

In this thesis I have tried to keep to the following stylistic and grammatical conventions:

- As far as possible I have tried to use the active voice in preference to the passive (e.g. "I developed a plugin" instead of "A plugin was developed") since I believe it has greater clarity in scientific writing, at the risk of sounding more colloquial. This is in line with the style guide of journals such as Nature[14] and Science.[15]
- I use the first person plural ("we", "our", "us") in two situations:
  1. When describing attitudes or approaches that are in common to those of us in Dr Douglas Armstrong's research group working in this area.
  2. When guiding the reader through an algorithm or mathematical derivation.
- In the main text of the thesis I refer to neuropil regions by their full names (e.g. "protocerebral bridge" instead of "pb"), but may use the abbreviated forms mentioned in the list on page 9 in figures, tables and captions.

In situations where images or graphs in this thesis may be unclear due to the limitations of page size, full resolution versions can be found online at `http://fruitfly.inf.ed.ac.uk/~mark/phd/` with other supplementary data, such as links to source code.

1.7.1 Nomenclature Issues

While I have been working on this thesis an international group consisting of experts on *Drosophila melanogaster* and wider insect neuroanatomy has been preparing a new standard nomenclature to describe regions of the fruit fly brain. The motivation for this effort is that some of the historical

[14] `http://www.nature.com/authors/author_services/how_write.html`

[15] `http://www.sciencemag.org/about/authors/prep/res/style.dtl`

terms that have been used for regions of the brain are either misleading or inconsistent with the homologous regions in other species.[16] In addition there are some distinguishable regions that have never been named. Unfortunately, the final revision of this new standard has not yet been published. As a result, the terminology is still subject to change, so I have avoided using terms suggested in early drafts or discussions, except where I know of no other name for the region.

An issue which is frequently raised in discussions about nomenclature is how some of the classical neuroanatomical terms, namely "rostral", "caudal", "ventral", "dorsal", "anterior" and "posterior", should be interpreted in the *Drosophila* brain where there are two different conventions: the neuraxis and the body axis. The former axis (more often used in literature on development) is curved within the adult brain, so in this thesis, which exclusively deals with adult flies, I will use instead the terms "superior", "inferior", "frontal" and "occipital" to refer to the body axis. The other terms may still arise in the text where they are part of a named neuropil region, such as the ventral bodies or the anterior ventrolateral protocerebrum.

[16] A similar effort to revise the nomenclature used to describe the avian brain [Reiner et al., 2004] has been very successful.

2 Data Acquisition

This chapter describes the methods used in acquiring the images of the central fly brain that are analysed in later chapters. The GAL4 enhancer trap system ([Brand and Perrimon, 1993], reviewed in [Duffy, 2002]), is used to direct expression of marker proteins to a small subset of the brain's neurons. These can then be detected via immunohistochemistry.

2.1 Selection of P{GawB} Lines

Our particular interest, as explained in the introduction, is connectivity of the fan-shaped body, so I selected a number of lines with interesting expression in that region as candidates for further study. These lines were selected from the online "Flytrap" archive[17] which shows the expression of the GAL4 driver lines in the Armstrong group's collection, derived from a screen of 1400 lines [Yang et al., 1995]. The particular driver insertion used in these lines is P{GawB} [Brand and Perrimon, 1993]. The lines I considered are listed in Table 1.

2.1.1 Mapping Insertions to Chromosomes

It was not recorded in the Flytrap database which chromosomes contained the 23y, 62y, 71y, c005, c061 and c159b insertions. These were discovered by myself and Dr Joanna Young, who was studying developmental neuroanatomy using an overlapping set of GAL4 lines. The chromosomes were worked out by the following procedure, supposing that the insertion is a line called ZZZ:

- Cross a ♂ ZZZ fly with ☿s of some balancer with *CyO*.
- If all the ♂ progeny from this cross are white-eyed, then the insertion is on the X chromosome.
- Otherwise cross a red-eyed, curly-winged ♂ from the progeny of that cross to CS w^-flies; if none of the red-eyed progeny have curly wings, the insertion is on the 2nd chromosome - otherwise it is on the 3rd chromosome.

(Note that there is a small chance that the insertion is on the 4th chromosome, which would not be detected by the mating scheme above. However, due to the comparatively small size of the 4th chromosome, this is unlikely.)

[17] `http://www.fly-trap.org`

Line Name	Chromosome	Insertion Location Known?	Comments
23y	2	No	
52y	?	No	Missing
62y	2	No	
71y	3	No	
104y	2L (26D)	No	
181y	2R (57B)	Yes - CG10543 (see [Armstrong et al., 2006])	
203y	X (1C)	No	Missing
210y	3L (70B)	No	
227y	X (1A)	Yes - CG3857 (see [Armstrong et al., 2006])	
c005	3	No	
c061	X	No	
c159b	2	No	
c255	3L	Yes - Gap1 *	
c259	?	No	
c546	?	No	Insertion Lost

Table 1 – P{GawB} lines with fan-shaped body expression which I initially considered for further study. (*Private communication from Dr Dean Baker.)

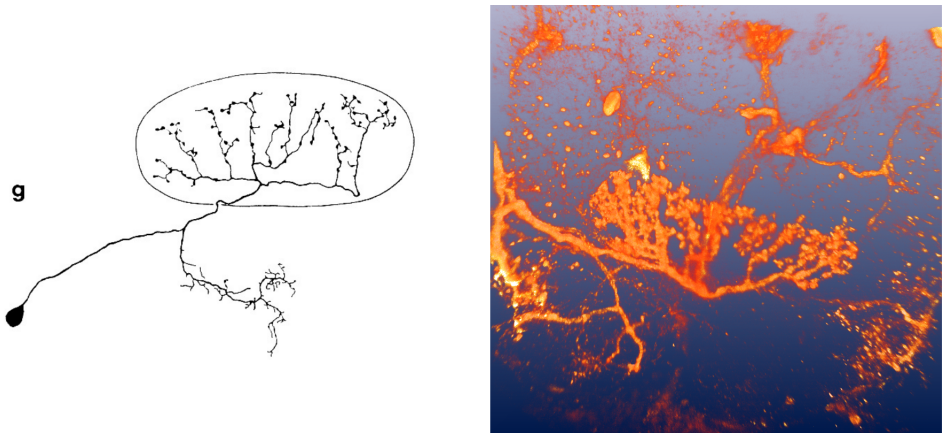


Figure 3 – On the left: an image copied from [Hanesch et al., 1989] showing a top-down view of the fan-shaped body and a large-field fan-shaped neuron; on the right: a projection of similar expression in the P{GawB} line 71y (rendered in Amira).

2.1.2 Narrowing Line Selection

It quickly became apparent that this was likely to be too many lines to consider given the period of study allowed for this PhD, and we made the decision to cut the focus down to 4 lines. c005, c061, 71y and 210y were chosen for further study based on the following reasons:

- c005: This line has a particularly sparse expression pattern with clear connectivity from the fan-shaped body to the superior and lateral protocerebrum.
- c061: As discussed in the introduction, this line appears to show a hypothesized connection between the fan-shaped body and mushroom bodies.
- 210y: Although the expression pattern of this line is far from sparse - it has expression in many cell bodies in the rind - in some scans there appears to be connectivity between the protocerebral bridge and the fan-shaped body. In addition there is expression in two distinct layers of the fan-shaped body as well as interesting mushroom body patterning.
- 71y: This is similar to c005 in many respects, but also has expression in the heel of the mushroom body. In addition, the expression in the superior layer of the fan-shaped body distinctly divides into processes in different segments, reminiscent of one of the neuron types described in [Hanesch et al., 1989]. (See Figure 3)

Ultimately, the decisions regarding which lines to preserve in such a study do have an arbitrary element: the expression patterns of the other lines have many interesting features too, of course. Hopefully we may be able to return to apply similar analysis to these lines in future work.

2.2 Crosses for Imaging

The flies which were dissected and imaged to create this image corpus were the progeny of virgin females from the P{GawB} line and males from the reporter construct, which was either the cell-filling marker UAS-lacZ or the membrane marking UAS-mCD8::GFP. The latter was designed to show finer detail, particularly in dense dendritic arbors, due to concerns that β-galactosidase (the protein encoded by the bacterial gene lacZ) did not penetrate into some of these regions. However, the staining of UAS-lacZ was found to be more robust, while still producing images with a great deal of connectivity information. Most of the images in this corpus are from flies expressing β-galactosidase, but there are some GFP-derived images as well.

2.3 Chemical Solutions

2.3.1 PBS

The phosphate buffer solution used in these experiments was at first prepared by the method described to me by Dr Dean Baker taken from *[J. Sambrook and E. F. Fritsch and T. Maniatis, 1987]* but later by dissolving Sigma Aldrich Phosphate Buffered Saline tablets (P4417) in distilled water.

2.3.2 Preparation of Paraformaldehyde

4% paraformaldehyde was prepared by following steps provided to me by Dr Dean Baker[18], which consist of dissolving 4g of paraformaldehyde in 100ml of PBS and afterwards adjusting the pH to 7.2.

2.3.3 PBT

The detergent solution, referred to as PBT, was made up of 0.3% Triton-X-100 in PBS.

[18]I owe a debt of gratitude to Dr Dean Baker, Jane Ewins and Dr Bilal Malik, who at various stages in this work agreed to look over my shoulder while I nervously prepared this solution.

2.4 Brain Preparation

All of the brains scanned for this body of work were prepared as described in this section. This is a protocol provided to me by Dr J. Douglas Armstrong and Dr. Dean Baker, based on one from Professor Heisenberg's laboratory in Würzburg, used in the StandardBrain [Rein et al., 2002] work.

1. Flies were pinned through the thorax and brains dissected out with Dumoxel forceps in 1x PBS at $4°C$.
2. The brains were fixed in 4% paraformaldehyde solution on ice for 30 minutes. (The fixation was begun not more than 40 minutes after dissection was begun.) A maximum of 3 brains were allowed per well.
3. The brains were washed in PBS for 5 minutes.
4. The brains were washed at least twice with PBT for 30 minutes.
5. The brains were incubated at $4°C$ overnight with the primary antibody.
6. The brains were washed three times with PBT for at least 30 minutes.
7. The brains were incubated at $4°C$ overnight with the secondary antibody.
8. The brains were washed at least twice with PBT, over the course of at least two days.
9. The brains were mounted in Vectashield on microscope slides and sealed with glycerol gelatin.

After this process the slides were kept in a box sealed from light, and scanned as soon as possible.

2.5 Reagents

2.5.1 Primary Antibodies

The following primary antibodies were used for this work:

anti-β-galactosidase: The anti-β-Gal antibody was acquired from ICN Pharmaceuticals. The antibody was used at a concentration of 1:1000.

anti-GFP: I used the rabbit anti-GFP antibody from Invitrogen / Molecular Probes, product number A11122. This antibody was used at a concentration of 1:400. It is worth noting that using the anti-GFP antibody to amplify GFP signal does cause a loss of information, as described in Professor Kei Ito's note on the subject [Ito et al., 2003]. In this section of work, however, our aim was to try to find as much connectivity as possible, so these former considerations were outweighed by this need. In section 7.1.2, however, I discuss a situation in which we have to be much more careful about this.

nc82 (anti-bruchpilot): The nc82 antibody was originally obtained from Dr Erich Buchner and latterly from the Developmental Studies Hybridoma Bank at the University of Iowa[19]. nc82 was used at concentrations between 1:10 and 1:20. The bruchpilot protein is expressed in active zones of neurons, and nc82 is the standard neuropil marking antibody used in adult *Drosophila* research [Wagh et al., 2006].

2.5.2 Secondary Antibodies

The secondary antibodies used in this work were all produced by Invitrogen[20]. Those used at various stages are listed below, with their respective concentrations:

- Alexa Fluor 488nm anti-Mouse IgG (at 1:400 in PBT)
- Alexa Fluor 488nm anti-Rabbit IgG (at 1:1000 in PBT)
- Alexa Fluor 488nm anti-Rat IgG (at 1:400 in PBT)
- Alexa Fluor 568nm anti-Mouse (at 1:400 in PBT)
- Alexa Fluor 546nm anti-Mouse (at 1:400 in PBT)
- Alexa Fluor 546nm anti-Rabbit (at 1:400 in PBT)

The nc82 antibody was detected with the 546nm anti-Mouse antibody and the β-Gal with the 488nm anti-Rabbit; in the few scans where anti-GFP was used to detect mCD8::GFP, I used the 488nm anti-Rabbit secondary antibody.

[19] `http://dshb.biology.uiowa.edu/`

[20] `http://www.invitrogen.com/`

2.6 Confocal Microscope Configuration

Since my interest was particularly in the central complex, I opted to use a 40x objective (namely the Plan-Apochromat 40x/1.0 Oil Iris), and to image only a region of the central brain that just included the elements of the central complex and the lateral extent of the mushroom bodies. The setup of our confocal microscope is shown in Figure 4. To avoid bleed-through, sequential tracks were used, interleaving illumination with the 488nm HeNe (Helium-Neon) LASER (with fluorescence detected at PMT1) and the 543nm Argon LASER (with fluorescence detected at PMT2).

2.7 Collected Data

The images I acquired which passed very basic quality controls are shown in Table 2 below, hereafter referred to as the "image corpus". The x, y and z spacing values are measured in μm and file sizes are measured in MiB (mibibytes). The "landmarks" column refers to the number of named central complex landmarks that could be found in that image - this process is explained later, in section 4.5.

2.8 Results

In the sections below I will briefly characterize the expression patterns that can be seen by eye in the scans from each of these four lines, but a more detailed discussion can be found in the results sections of Chapter 5 (describing the connectivity) and Chapter 6 (describing expression within the fan-shaped body). A maximum intensity projection through the xy planes of four representative registered scans from these lines can be seen in Figure 5, in which the more prominent features can be seen. (More detailed views of 210y and 71y can be also seen in Figures 6 and 7.) Some of these notable features are labelled and further described below, but those which are more difficult to pick out can be examined via the links to the images at: `http://fruitfly.inf.ed.ac.uk/~mark/phd/`

2.8.1 210y Expression

The strongest expression in 210y scans is noticeable throughout the mushroom bodies. The peduncles, alpha lobes and gamma lobes all have strong expression running throughout their lengths. ("mb" in Figure 6 D.) In the case of the peduncles and alpha lobes this is around the outside of the structures, as if wrapping them. Expression in the fan-shaped body appears to be in two strata, some in a deep layer at the superior end, and some slightly stronger expression in one of the most inferior

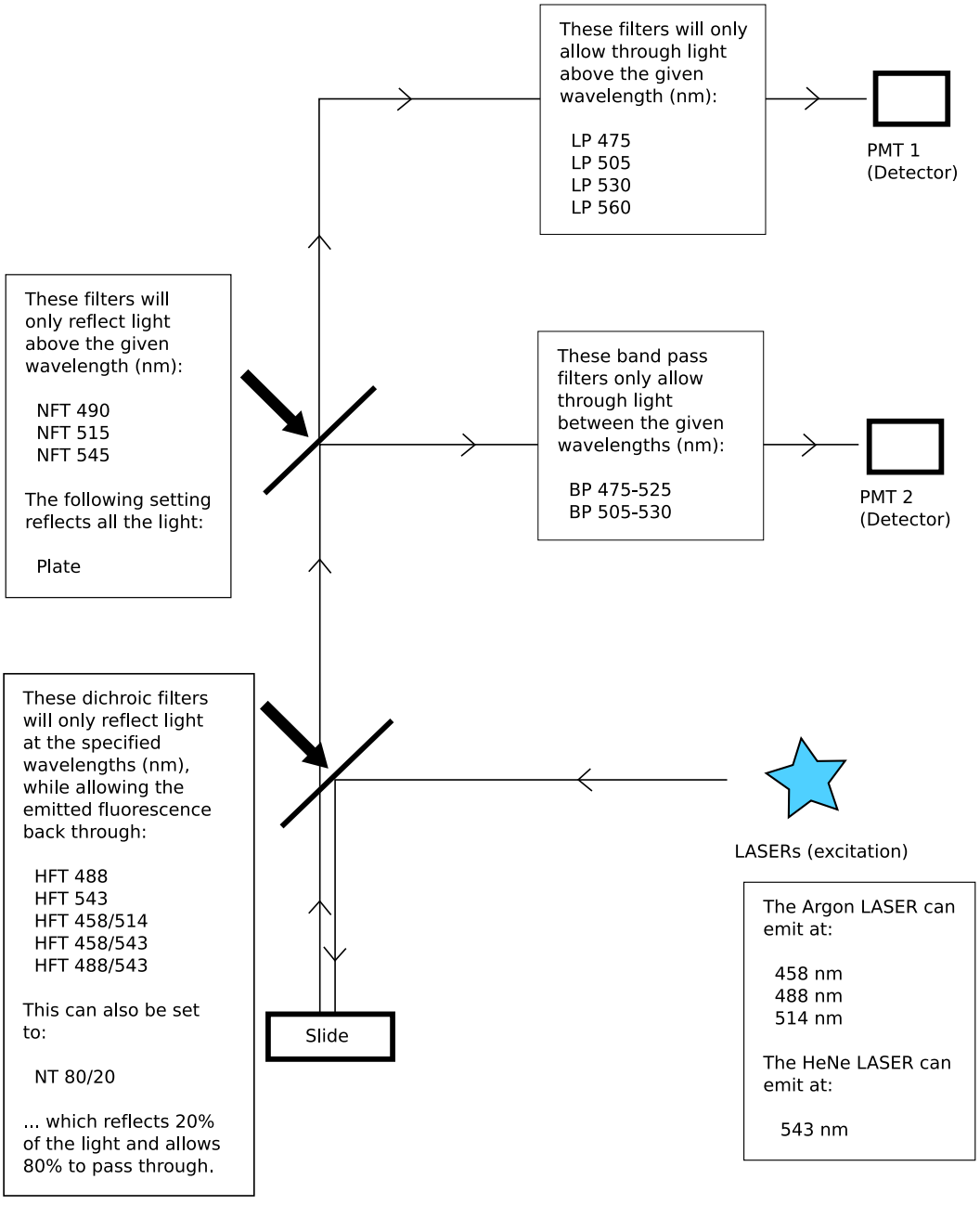


Figure 4 – A diagrammatic representation of the available filters and mirrors in the Armstrong group and Jarman group's confocal microscope.

Filename	Line	Reporter	Width	Height	Depth	x Spacing	y Spacing	z Spacing	Landmarks	FileSize
210y-40x-central-complex-CA.lsm	210y	UAS-lacZ	1024	1024	90	0.27	0.27	1.2	9	184
210y-40x-central-complex-CB.lsm	210y	UAS-lacZ	1024	1024	93	0.27	0.27	1.2	8	190
210y-40x-central-complex-CD.lsm	210y	UAS-lacZ	1024	1024	102	0.22	0.22	0.9	9	208
210y-40x-central-complex-CE.lsm	210y	UAS-lacZ	1024	580	110	0.25	0.25	1	8	127
210yAC.lsm	210y	UAS-lacZ	800	800	149	0.29	0.29	1	9	188
210yAD.lsm	210y	UAS-lacZ	800	800	141	0.29	0.29	1	9	178
210yAE.lsm	210y	UAS-lacZ	800	800	109	0.29	0.29	1.2	9	138
210yAO.lsm	210y	UAS-lacZ	1024	1024	113	0.17	0.17	1	9	231
210yAP.lsm	210y	UAS-lacZ	512	512	143	0.39	0.39	1	9	78
mhl-middle-ish(onlygoodoneon(E))210yxUAS-lacZ(0).lsm	210y	UAS-lacZ	900	900	135	0.26	0.26	1	9	214
71yAAeastmost.lsm	71y	UAS-lacZ	800	800	93	0.29	0.29	1.2	9	117
71yABwestmost.lsm	71y	UAS-lacZ	800	800	114	0.29	0.29	1	9	144
71yAF.lsm	71y	UAS-lacZ	800	720	92	0.25	0.25	1.2	9	104
71yAM.lsm	71y	UAS-lacZ	800	800	121	0.29	0.29	0.98	9	153
71yAN.lsm	71y	UAS-lacZ	1000	700	100	0.23	0.23	1.32	9	136
71yAQ.lsm	71y	UAS-lacZ	800	800	131	0.29	0.29	1	9	166
71yAR.lsm	71y	UAS-lacZ	800	800	117	0.29	0.29	1	9	148
71yAS.lsm	71y	UAS-lacZ	800	800	101	0.29	0.29	1.2	9	128
71yAT.lsm	71y	UAS-lacZ	800	800	101	0.29	0.29	1	9	128
mhl-71yxUAS-lacZ(0).lsm	71y	UAS-lacZ	900	900	139	0.26	0.26	1	9	221
c005BA.lsm	c005	UAS-lacZ	800	800	117	0.29	0.29	1	9	148
c005BB.lsm	c005	UAS-lacZ	800	800	129	0.29	0.29	1	8	163
c005BC.lsm	c005	UAS-lacZ	800	800	118	0.29	0.29	1	8	149
c005BD.lsm	c005	UAS-lacZ	800	800	125	0.29	0.29	1	9	158
c005BE.lsm	c005	UAS-lacZ	800	800	121	0.29	0.29	1	9	153
c005BF.lsm	c005	UAS-lacZ	800	800	107	0.29	0.29	1	9	135
c5xUAS-CD8GFP-24x-cc-BD.lsm	c005	UAS-mCD8::GFP	1024	1024	103	0.22	0.22	0.8	9	210
c5xUAS-CD8GFP-40x-central-complex-BE.lsm	c005	UAS-mCD8::GFP	1024	1024	106	0.22	0.22	0.8	9	217
c5xUAS-CD8GFP-40x-central-complex-BF.lsm	c005	UAS-mCD8::GFP	1024	1024	101	0.22	0.22	0.8	9	206
c5xUAS-CD8GFP-40x-central-complex-BG.lsm	c005	UAS-mCD8::GFP	1024	1024	115	0.22	0.22	1	9	235
c5xUAS-lacZ-40x-cc-BA.lsm	c005	UAS-lacZ	800	800	97	0.29	0.29	1	6	122
c5xUAS-lacZ-40x-cc-BB.lsm	c005	UAS-lacZ	800	800	93	0.29	0.29	1.1	8	117
c5xUAS-lacZ-40x-cc-BC.lsm	c005	UAS-lacZ	1024	1024	100	0.22	0.22	1	9	204
mhl-middle(C)c5(0).lsm	c005	UAS-lacZ	512	512	90	0.45	0.45	1	8	49
mhl-westmost(D)c5(0).lsm	c005	UAS-lacZ	800	800	107	0.29	0.29	1.2	9	135
c061AG.lsm	c061	UAS-lacZ	800	720	117	0.29	0.29	1.2	9	133
c061AH.lsm	c061	UAS-lacZ	800	720	116	0.29	0.29	1.2	9	132
c061AI().lsm	c061	UAS-lacZ	800	800	111	0.29	0.29	1	9	140
c061AJ.lsm	c061	UAS-lacZ	512	512	103	0.37	0.37	1	9	56
c061AK.lsm	c061	UAS-lacZ	800	800	116	0.29	0.29	1	9	147
c061AL.lsm	c061	UAS-lacZ	800	800	121	0.29	0.29	1.03	9	153
c061AU.lsm	c061	UAS-lacZ	800	800	119	0.29	0.29	0.8	8	150
c061AV.lsm	c061	UAS-lacZ	800	800	123	0.29	0.29	1	9	155
mhl-eastmost(A)c61(0).lsm	c061	UAS-lacZ	800	800	106	0.18	0.18	1.2	6	134
mhl-northernmost(A)c61(0).lsm	c061	UAS-lacZ	800	800	83	0.24	0.24	1.2	5	105
mhl-theotherone(A)c61(0).lsm	c061	UAS-lacZ	800	800	102	0.29	0.29	1.2	9	129
mhl-westmost(B)c61(0).lsm	c061	UAS-lacZ	1024	1024	128	0.22	0.22	1	9	262

Table 2 – Details of images in the original central complex image corpus.

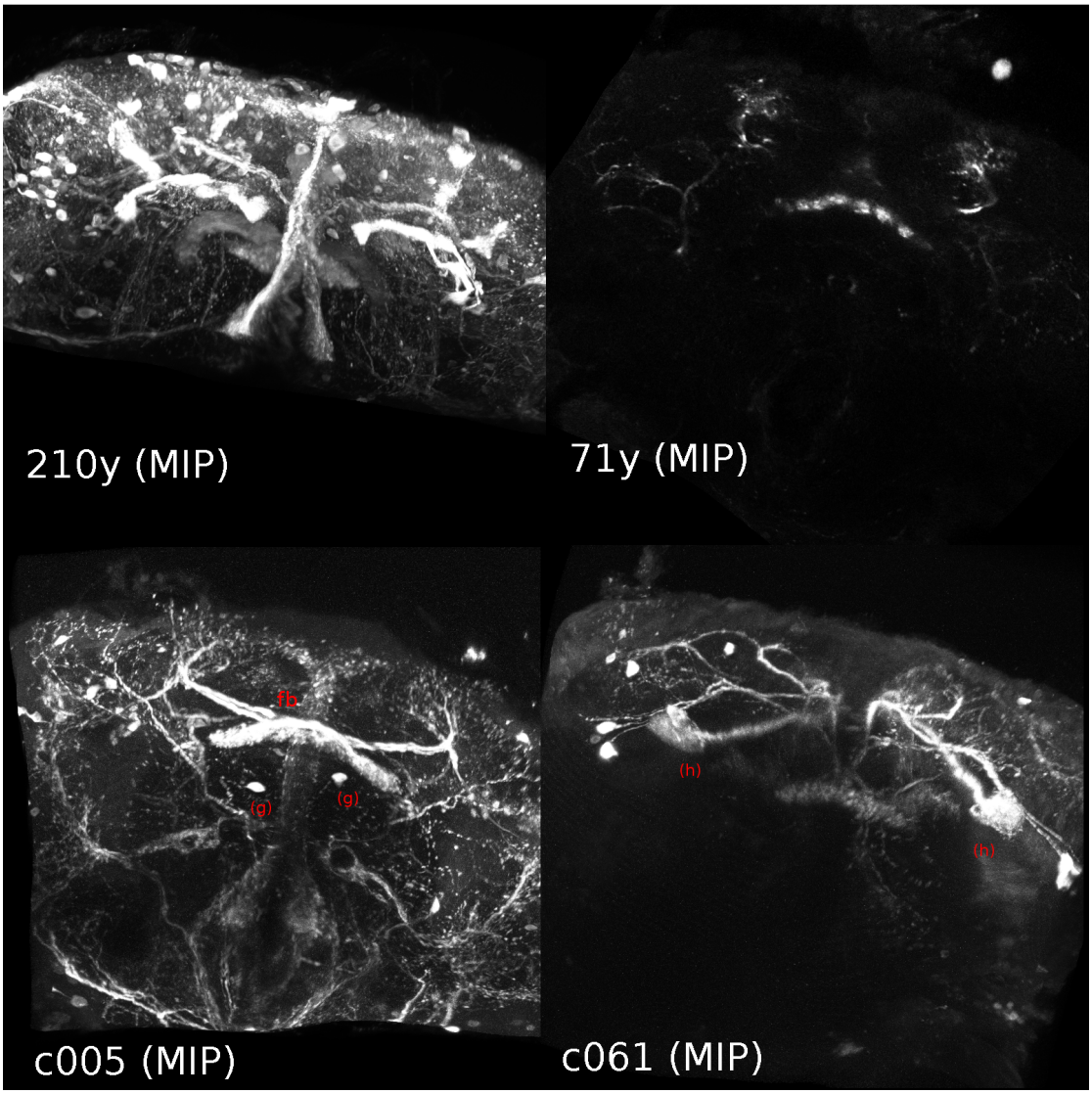


Figure 5 – MIP renderings of four good quality scans of each line. For ease of comparison these have all been registered into the same space using the CMTK method described in Chapter 4.

layers. ("fb (sup)" and "fb (inf)" in Figure 6 B.) The protocerebral bridge clearly shows expression completely filling the region, but at a lower expression level than that in the fan-shaped body or mushroom bodies. ("pb" in Figure 6 A.) The median bundle is prominently visible, with expression going around the œsophagus below. (Figure 6 D.) Further discussion can be found in section 5.5.3.

The other very obvious feature of 210y scans is that large numbers of cell bodies are visible on the occipital side of the brain, particularly around the calyces of the mushroom bodies. There are other clusters of cell bodies around the lateral horn. (Figure 6 C.)

2.8.2 71y Expression

The 71y line picks out a narrow layer of the fan-shaped body, but not quite at the superior edge - this separates into nearly distinct branches towards the protocerebral bridge. ((a) in Figure 7 B.) Just superior to the fan-shaped body is a bright chiasma with processes leading to the fan-shaped body layer and the superior protocerebrum. ((b) in Figure 7 B.) There is clear expression in the spurs of the mushroom bodies and in the core of the peduncles of the mushroom bodies near to the spurs. ((c) in Figure 7 A.) However, this is not seen throughout the peduncles; it becomes undetectable towards the calyces.

There are a few small clusters of cell bodies visible:

- A group which may either be very large cell bodies or several adjacent to each other can be found superior to the sub-œsophageal ganglion and lateral to a region sometimes known as the posterior lateral protocerebrum. ((f) in Figure 7 A.)
- A couple of cell bodies can be found on each side just lateral to the antennal lobes, between them and the anterior ventrolateral protocerebrum. ((e) in Figure 7 A.)
- A few are visible superior to the protocerebral bridge.
- Some cell bodies are also found just lateral to the calyces on both sides.

Further discussion can be found in section 5.5.5.

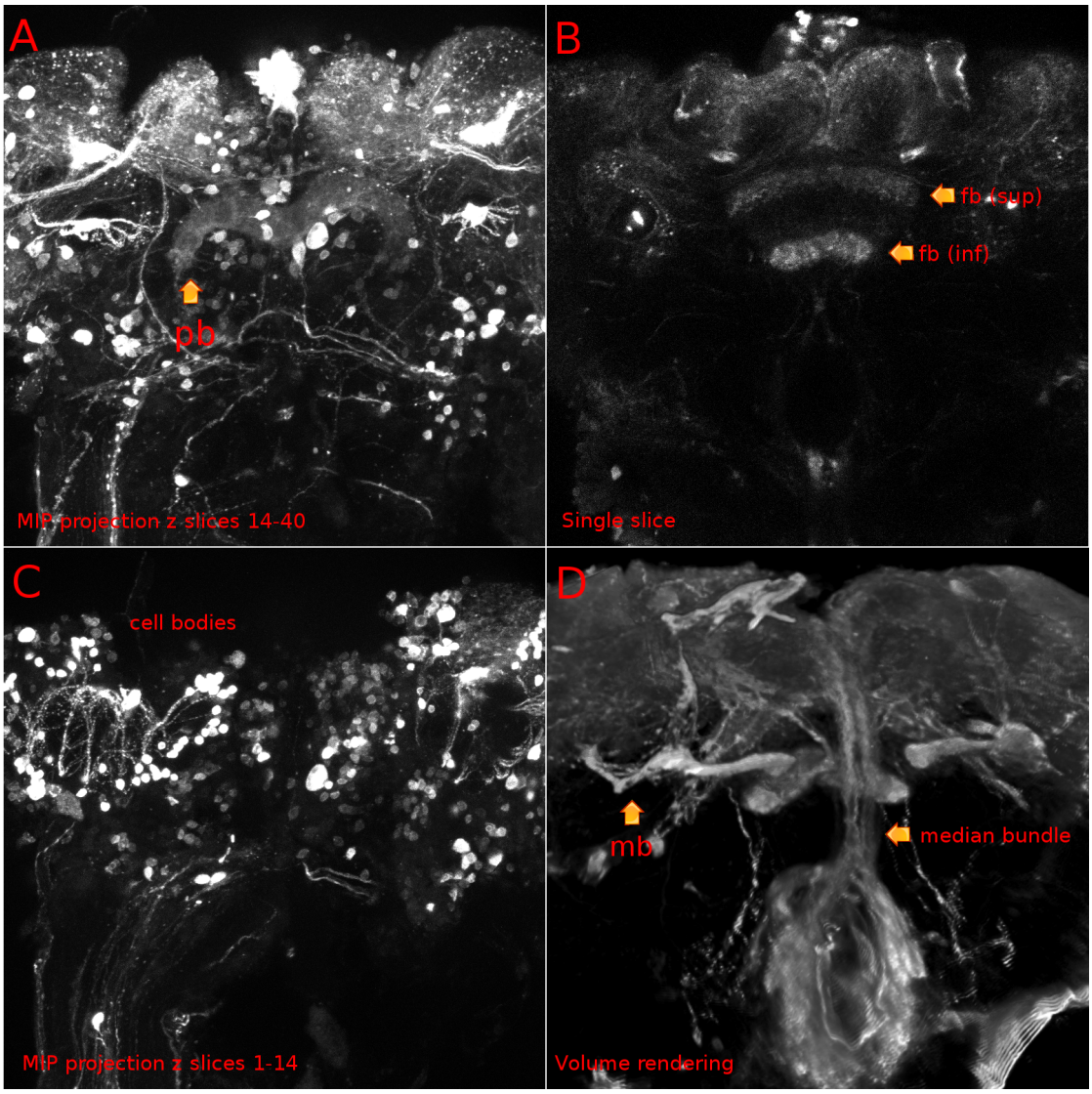


Figure 6 – Various renderings of 210y, in particular the image 210yAC.

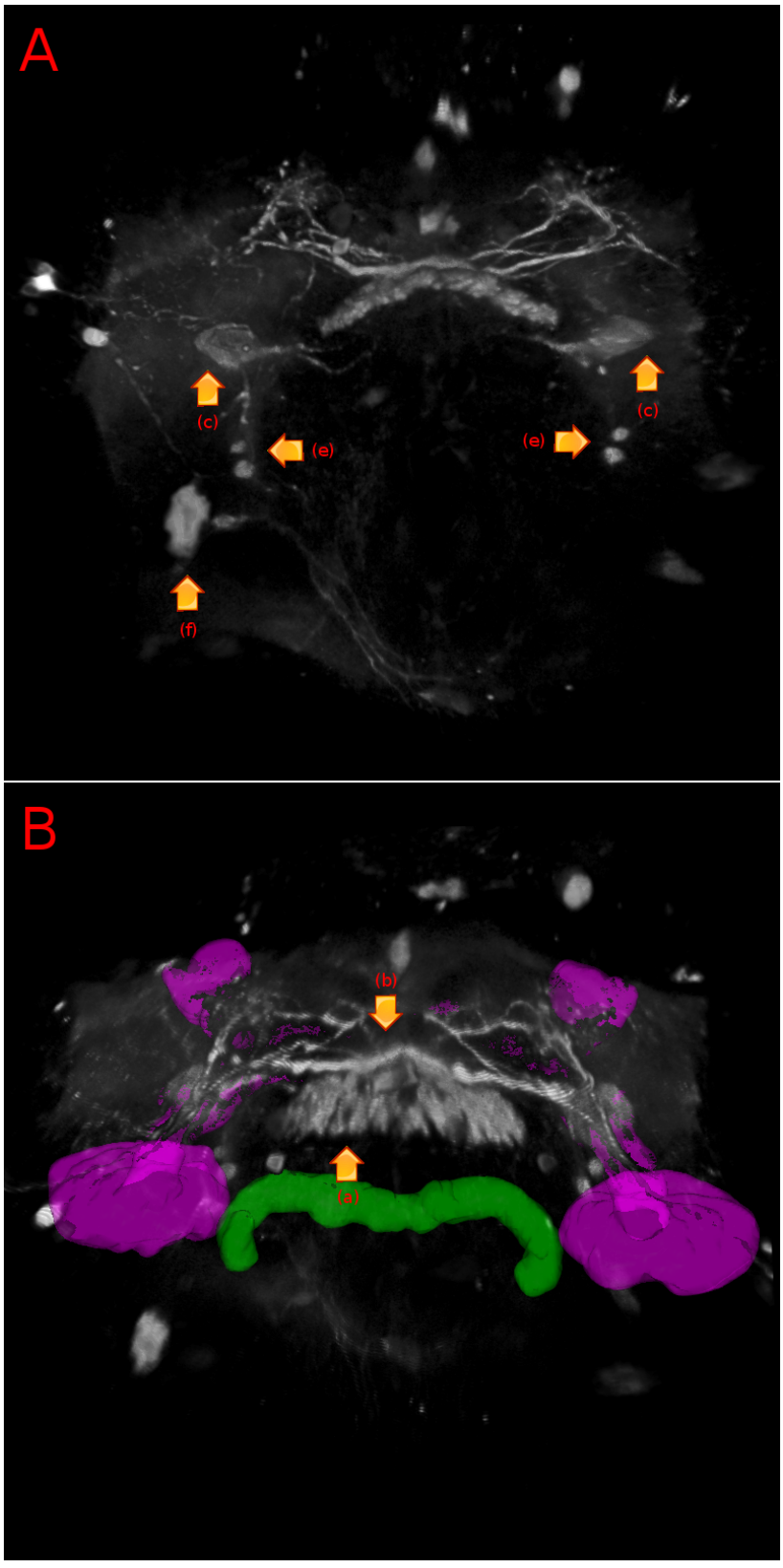


Figure 7 – A. Volume rendering of 71y from the frontal side of the brain; B. A volume rendering of 71y from the occipital side. The neuropil regions of the protocerebral bridge (green) and the mushroom bodies (magenta) have been added to B in order to make the relative position of the other features clear.

2.8.3 c005 Expression

The most prominent feature of the c005 line's expression is in the superior layers of the fan-shaped body, which led to its use in previous work on that structure [Liu et al., 2006]. Clear bundles of neurons lead to this apparently contralaterally from the superior side of the fan-shaped body. ("fb" in Figure 5.) The cell bodies are distributed around the rind, but particularly noticeable in the following regions:

- Clusters just lateral to the calyces of the mushroom bodies;
- A pair just medial to the calyces of the mushroom bodies, between the mushroom bodies and protocerebral bridge;
- At least two occipital to the protocerebral bridge ((g) in Figure 5);
- A cluster around the sub-œsophageal ganglion;
- Low expression through the rind around the antennal lobes.

The network of processes visible with c005 is rather complex, and shown more clearly in section 5.5.2. There are neuronal processes around the pedunculus of the mushroom body (the posterior ventrolateral protocerebrum) and a neuronal arbor in the anterior ventrolateral protocerebrum. There are processes that appear to connect these regions to the sub-œsophageal region. There are fibres passing down the median bundle to the œsophagus, and expression surrounding that organ. There is punctate expression in processes running along the great commissure. Processes pass from the great commissure towards the optic lobes just occipital to the antennal lobes. There are horizontal processes running in a tract just above the sub-œsophageal ganglion.

2.8.4 c061 Expression

c061 shows strong expression in the spurs of the mushroom bodies. Fan-shaped body expression is limited to one of the lower layers, not quite the most inferior. Around the mushroom bodies' gamma and beta lobes (in a neuropil region sometimes referred to as the crepine) there is some expression typical of neuronal arbors. Similarly, there is a network of processes that passes around the alpha lobes of the mushroom bodies.

There is some low level expression seen through the cortical rind, but there are very clearly marked cell bodies lateral to the calyces of the mushroom bodies with processes leading from them to the superior protocerebrum.

A network of processes is visible around the œsophagus, and some expression in the commissure above the sub-œsophageal ganglion, as in c005.

Further discussion can be found in section 5.5.4.

2.8.5 Summary

The data collected via the methods described in this chapter were of an acceptable quality in that the signal from the GAL4 channel was clear enough to pick out neuronal processes representing the type connectivity data that we are interested in, such as fan-shaped neurons. The quality of the nc82 channel in some of these scans was still rather poor, since this antibody is more difficult to optimize for than anti-GFP or anti-β-galactosidase. I excluded those in which the central complex regions were not identifiable, but still left in some where they appeared somewhat distorted since it was possible that these could be correctable via elastic registration. In some cases this turned out not to be the case, so in Chapter 4 some further brains are excluded where the registration failed - these failures are typically attributable to poor image quality.

3 Confocal Image Data Archiving

It became clear at an early stage of this project that a major challenge was going to be the management of large amounts of confocal image data. This led to the development of an online system for archiving, viewing and annotating image stacks. The development and evaluation of this tool is described in this chapter.

3.1 Why are confocal images problematic?

The two major reasons that managing the image stacks is problematic are:

1. The images are typically hundreds of megabytes in size.
2. The information in the image data is very expensive to collect.

That the first point is true should be clear, but the practical implications of it are that one cannot afford to keep multiple redundant copies of the images on desktop computers and that moving the data from one system to another is best done with portable USB hard disks unless there is a fast network connection available.

On the second point, it is easy to forget how expensive these data are to generate, since this cost involves:

- The time involved in preparing, mounting and scanning the brains;
- The cost of the reagents needed for the immunohistochemistry;
- The running costs of the confocal microscope (which in our laboratory's system is dominated by the cost of replacing the mercury lamp and argon LASER).

As a result, it is very important that these images are archived in such a way that they are unlikely to be damaged. In this case, what I consider to be damage is not just limited to the usual considerations such as accidental deletion, theft, media corruption, etc. but also includes any manipulation of the file that will change the data as it was stored at acquisition, e.g. by annotation in viewing software. This point can be critical, since a great deal of information about the acquisition is stored in the image metadata and it is not easy to guess in advance which such information might be needed in

subsequent analysis. (For example, in the course of this body of work, I discovered at a late stage that it would be useful to analyse the PMT settings for each image channel - since the original data was carefully preserved, this was simple to extract.)

Anecdotally, experience has also shown that within many research groups that deal with large amounts of biological data, approaches where each user has to take a non-trivial[21] action to back up their data have a greater susceptibility to disastrous data loss.

3.2 System Requirements

In discussion with my supervisors, I decided that the system that we wanted should have at least the following properties:

1. The user interface to the archival system should be a web front-end, to encourage use by other members of the group and mostly eliminate cross-platform development time.
2. Users must be able to upload images either singly (using a web form) or in batches using an uploader tool.
3. The system should be able to parse metadata from Zeiss LSM files.
4. There must be some positive benefits that the users will immediately get from uploading their data to the system, quite apart from the "worthy-but-dull" bonus of reliable backups.
5. The system should not allow exact duplicate images to be uploaded.
6. The system should allow simple off-site back-ups of the image data.
7. The web front-end should provide a simple and universal way of sharing confocal data within a research group and selectively to people outside.
8. The originally uploaded data should be regarded as sacrosanct and never changed.
9. The system should provide a simple way to allow students on short-term projects to work on our archive of images.
10. The system should be easy to install on any modern Linux distribution, and work on a single machine.

[21] In fact, some might say "any".

It became apparent that a very simple design could achieve all of these goals, with a very low cost in development time. This was largely because at an early stage of my PhD I had ported much of the ImageJ LSM_Reader plugin[22] to C++ and written a small suite of programs for extracting data from 8-bit LSM files. In particular it could extract the metadata about the image acquisition and convert image stacks to a series of PNG files.

This first implementation did meet the design goals, and was favourably received, which encouraged us to plan for an improved system. The key aspects of this earlier version that allowed us to meet the goals above were:

- The MD5sum of the uploaded scans were used as a primary key. This also allowed us to simply reject attempts to upload duplicate scans.
- Once uploaded, the image files were copied to a location on the web server such that they could be served as a static file from the web server under a path that includes the MD5sum. Serving these as static files avoids the serious performance hit from returning large files as dynamic content, while needing to know the MD5sum in order to download the scan provides a level of security, since this value is effectively unguessable.
- The features alluded to in point 4 were essentially:
  - The incorporation of the MipJ applet for doing 3D projections of the confocal stacks in a Java applet. This was developed by Douglas Cowan, a student supervised by Douglas Armstrong, as part of his MSc project in 2003. It was easy to incorporate this into the database with a few simple changes to the source code.
  - A simple Javascript viewer for quickly looking through a stack, without the problems associated with Java applets.
  - The system provided a simple way of showing images to other people in the group who were working remotely.

These features were all ones that I wished to preserve in future versions of the system. Some ideas we experimented with turned out to not be so successful, however. In particular, I wished to use

[22]This plugin was written by Patrick Pirrotte, Yannick Krempp and Jerome Mutterer. It has now been superseded by LSM_Toolbox, which can be found at `http://imagejdocu.tudor.lu/Members/ppirrotte/lsmtoolbox`

the Java applet version of ImageJ in order to provide a more sophisticated in-browser viewer of the confocal scans. This turned out to have a number of usability problems, however:

- There would be a large wait during which the browser was unresponsive while the scan was downloaded to the client.
- Despite there having been many years for the interfaces between browsers and the Sun Java plugins to mature, I still experience frequent crashes on loading applets.
- It is not obvious to the user what is actually happening when viewing an image in the ImageJ applet, i.e. that the scan has been downloaded locally but is essentially inaccessible except in the applet.
- Viewing scans in the appletized ImageJ requires a large amount of Java heap space to be allocated, and the instructions for changing this[23] are complex for many users.

As a result, this was abandoned for the future versions. Interoperability with ImageJ on the client was still maintained, but by allowing plugins in the locally installed ImageJ to upload and fetch annotations from the archive using the API.

3.3 Planning

In planning for a new version of this confocal image archive, we considered whether this effort would be largely redundant because of the existence of some established and mature systems for archiving biological image data. The two systems that we have to bear in mind are developed between a number of academic institutes under the name of the Open Microscopy Environment. These alternatives are:

- The OME server (`http://www.openmicroscopy.org/site/documents/data-management/ome-server`)
- OMERO (`http://www.openmicroscopy.org/site/downloads/omero-downloads`)

While the features of these two systems overlap to a certain extent, we would not wish to deploy OMERO since it requires a Java-based client, and one of our usability requirements is that the

[23]The -Xmx parameter must be added in the control panel:
`http://java.sun.com/j2se/1.4.2/docs/guide/plugin/developer_guide/control_panel.html`

system should use a standard web browser as its client. The OME server, however, has a similar design philosophy and feature set as our ideal system. The reasons that we were inclined to pursue the development of our own system instead of using OME server included the following:

- We believed that the system we required was essentially a simple one, and that the required development time would be short, as demonstrated by the prototype's development. The Open Microscopy Environment project is a large one both in terms of the number of developers involved and its comprehensive feature set. As development of our system progressed, however, this reasoning may not have remained valid. It may well, in retrospect, have been a better investment of time to adapt the OME server to meet our additional requirements. However, there are some unique features of our current system (such as the ImageJ-based job server, Javascript stack viewer, landmark annotation, MIP projection generation) which nonetheless distinguish it from any other comparable systems.

- The OME server can preserve a copy of the source files for archival purposes but in general works on copies of the image data converted to the OME-TIFF format. This approach has a number of advantages, but was one that we resisted due to the enormous problem of trying neither to lose nor misinterpret data when converting from one of the proprietary confocal microscope formats into a universal file format such as OME-TIFF. In addition, the universal file format approach typically doubles the storage space required for the image data if the original data must be preserved.

- One of the trickier parts of developing an online microscope scan interface is to make a slice-by-slice stack viewer work well in a web browser. The OME server's viewer is based on SVG, which at the time was not well supported in many web browsers, and on some platforms required an additional plugin to be installed. My strong preference was to use a more basic Javascript-based user interface for this which did not use SVG for rendering, but instead manipulated HTML.

The OME server does have a much wider set of features than we planned for our system, however. In particular, we did not intend to provide special facilities for dealing with time series or file formats that are made up of multiple files per image. (Both of these limitations can be worked around, but not particularly elegantly.)

In summary, our intention was to develop a system that would be smaller and simpler than the OME server, but which would still provide a useful solution for sharing, backing-up and annotating

confocal data. The extent to which we achieved this goal is discussed further in the results section of this chapter.

As far as I am aware, there are few other publically available systems that provide a similar feature set to either our proposed system or the Open Microscopy Environment. We know through personal communication that many other groups similarly have developed in-house solutions, but I know of none of these that are made available to the community at large. There do exist commercial equivalents, such as Biotrue's system,[24] but we have not considered servers of this type where the source code is not available under a Free Software (or "Open Source") license; it is important that we should be able to develop the system to accommodate further server side development and image processing.

3.3.1 Restrictions in the Prototype and Proposed Solutions

Perhaps the most serious restriction of the prototype system that I developed was that it would only support 8-bit LSM files. This happens to be the format that almost all the images generated by our group are saved in, but it was a serious limitation for use by anyone else. In particular, most of the image processing tools that we use cannot *write* LSM files, only read them. In addition, writing image loaders is time-consuming, and it was out of the question for me to personally write new ones for each new image format we wished to support. We could, instead, insist that before files are uploaded to the system they should be converted to some common format, but this both produces usability problems and breaks our most important principle, i.e. that the data stored in the archive should be as acquired from the microscope. My proposed solution for this was to use ImageJ on the server side to process the uploaded files, instead of my custom C++ loader. The ImageJ development community has written loaders for an enormous range of image formats currently in use in medical and biological imaging - indeed, I have not yet come across any such file format in the course of my research that does not have an ImageJ loader.

The second key limitation was that if data needed to be generated in order to view a particular page (for example, unpacking the image stack into a sequence of PNG files) that data would be generated as the page loaded. In other words, if the operation to generate the data was lengthy, or the server was particularly loaded at the time, the user experience would be of an extremely long page load

[24] `http://www.biotrue.net`

time, and some proxies or browsers might time-out the request. My proposed solution for the new version was to have a job queuing system on the server side. The implementation of this is discussed further in the next section.

3.4 Current Architecture

3.4.1 Design of the Current System

A diagrammatic representation of the current confocal image archive system is shown in Figure 8. This may not necessarily make sense in isolation, but each component is described and justified in the following sections, and it may be useful to refer back to this diagram to understand the relationship between each component.

3.4.2 Network Shared Disk Storage

In terms of basic storage, the heart of the system should be a large amount of network attached storage that can be mounted as a single directory from the machine running Apache (the front-end) and those running the ImageJ job servers (the back-ends). In our current installation these are all running on a single system with a terabyte of disk storage, so no network file systems are necessary, but in a higher capacity system we would expect these to run on different systems mounting the network attached storage over NFS, ZFS, AFS, or some similar distributed filesystem protocol.

An interesting consideration of scalability in this kind of web service is that the most expensive dimensions along which to scale up are CPU and RAM. Network bandwidth and disk storage are relatively easy to scale: with most ISPs it is economical and administratively simple to buy more bandwidth when necessary and vast amounts of new disk storage can be easily added to the network at little performance hit. (Nowadays gigabit ethernet gives us bandwidth greater than many types of locally-attached storage that are still widely used.[25]) The point of this is that the simplest way of being able to scale up CPU is to be able to add more hosts as back-ends, so this has been incorporated into the design from the start.

[25] e.g. Ultra-wide SCSI; more recent buses such as Serial ATA, of course, still have better performance than iSCSI over gigabit ethernet `http://en.wikipedia.org/wiki/List_of_device_bandwidths` on 2009-09-10.

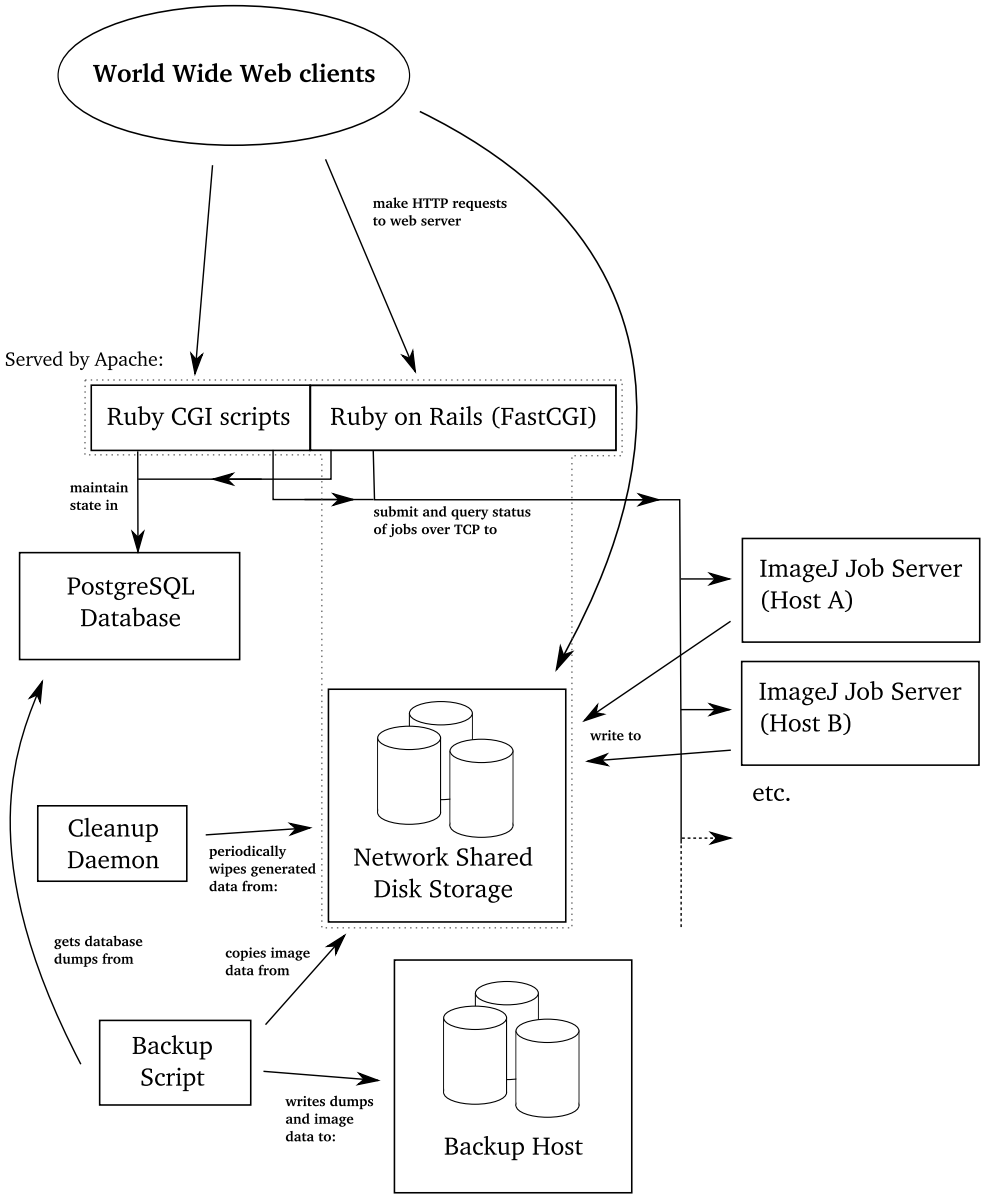


Figure 8 – The current architecture of our web-based confocal image archive.

3.4.3 ImageJ-based Jobs

The model of a server side job in this system is very simple. It must meet the following restrictions:

1. The job must be implemented as an ImageJ macro.
2. The job must (and must only) output to a particular directory passed to the macro. (The output directory will already have been created by the time the macro starts.)
3. The job indicates its progress and its completion by writing to special filenames in its output directory.
4. On any error the macro must throw a RuntimeException.

With regard to point 1, ImageJ macros can call plugins and perform effectively arbitrarily complex image processing, so this imposes no particular restriction on the type of analysis that can be conducted in a server side job. On the contrary, developing new macros for doing server side jobs is a very fast process, and, as such, the platform provides a very easy way to deploy interesting image analysis protocols to users who would not normally be able to run them. (Typically, this would be because some jobs are so expensive in terms of CPU time that they cannot be conveniently run on desktop computers.) There are a number of restrictions that this policy also introduces, but they are largely irrelevant for the type of processing we expect to be done via this system - for example, fork-and-exec and many other standard POSIX facilities are impossible to access with pure Java.

The second and third points specify the only way in which jobs can communicate back to the rest of the system. This is restrictive in one important respect: the clients of the job server must discover progress towards completion by polling. This mechanism does, however, have the advantage that this information can be discovered by any other component by reading the contents of the output directory - language interfaces and network protocols are not an issue. The various files that the job can write to the directory are as follows:

- On successful completion, the image macro should create a zero-byte file called "`.completed`" in the directory.
- During operation the macro may write an US-ASCII representation of the proportion of the job that has been done to a file called "`.progress`", e.g. a file of the 4 octets `0x30 0x2E 0x32 0x35` ("0.25") would indicate that the job is a quarter complete.

On the fourth point, any exceptions that are thrown by the macro continue up to the job server on that host. Most of the macros we use at the back-end just immediately call an ImageJ plugin, and ImageJ plugins cannot propagate checked exceptions upwards due to there being no "throws" clause in the run method of the PlugIn interface. As a result, we specify that these must be of class RuntimeException, i.e. a type of "unchecked" exception in Java.

3.4.4 The ImageJ Job Server

There is one ImageJ job server per back-end host. The Ruby CGI scripts and Ruby on Rails web front-end communicate with these over a simple protocol over TCP. Each request is authenticated using a challenge-response system based on a different shared secret between the front-end and each back-end. This should stop an attacker from sending malicious commands to any back-end. In the future, however, it would be better to use SSL or SSH wrapped communication between the front- and back-end.

The JobServer listens on port 2061. The protocol consists of a client writing a single `\r\n` terminated line of tab-separated fields and the server sending back a similarly formatted line. The command that the client sends is specified in the first field, and must be one of the following:

- If the command is "`start`" then there may only be two other fields (thus three in total). The first is an ImageJ macro expression, while the second is an estimate of an upper bound of the number of mibibytes that the command requires to run. For example, a complete command which is expected to complete not using more than 24MiB of memory might be:

  ```
  start\trun('Example Plugin','[filename=/archive/data/2008-08-14/c020dccf76c60\
  b2b5e187624cfa31a8f/scan.bin] directory=[/archive/data/2008-08-14/c020dccf76c\
  60b2b5e187624cfa31a8f/example-output/]');\t24\r\n
  ```

  The server will return a line with two fields, the first being "`started`" and the second being a job ID. These job IDs are unique to a single job server, as distinct from "database job IDs" (described below) which are unique across a complete installation of the system.

- If the command is "`query`" then there may only be one other field (thus two in total) which is a job ID as returned from a "`start`" command. The response may be one of the following:

  - If the first field of the reply is "`finished`" this will be the only field. This reply indicates that the job has completed successfully.
  - If the first field of the reply is "`failed`", there will be one further field containing an error message. This indicates that the job has failed for the reason given.
  - If the first field of the reply is "`working`", there will be two further fields (thus three in total). This indicates that the job is currently being performed. The second of the fields will be a decimal representation of the proportion of the job that has been completed. The third is typically empty, but may be an estimate of the completion time in one of the ISO 8601 suggested formats, YYYY-MM-DDTHH:MMZ (e.g. "2008-08-18T08:40Z").
  - If the first field of the reply is "`queued`", there will be one further field (thus two in total). The other field will be an integer representing the number of jobs currently ahead of this one in the queue, or -1 if the job has been so recently started that it is not yet enqueued.
  - If the first field of the reply is "`unknown`" this will be the only field. This reply indicates that the job ID was unknown.

- If there is any error (for example if the command sent by the client is unknown, or the command sent has the wrong number of fields) then the reply from the server will be made up of two fields, the first being "`error`" and the second being a descriptive error message.

The handling of errors by this job server is a difficult business. The server itself is an ImageJ plugin launched in a modified version of Fiji (see section 1.6.2) running in the headless mode developed by Dr Johannes Schindelin. The most important modifications remove all of ImageJ's exception handling when invoking macros and plugins, so that exceptions can be caught and reported by the server. The problem with this approach is that it is not uncommon when processing large images for OutOfMemoryError errors to be thrown by the plugins. These errors can be caught with a "`catch( Throwable t )`" but with a number of caveats. Sun's documentation consistently discourages catching OutOfMemoryError in this way, but does not make it completely clear whether it is possible to handle these correctly in practice. Many Java applications (e.g. ImageJ, JEdit) do handle OutOfMemoryError, sometimes using hacks such as allocating a "hedge"[26], but if the

[26] A block of memory which is allocated on startup simply so it can be freed in the case of OutOfMemoryError in order to improve the chances of recovering

allocation that fails is not in the expected thread, but, for instance, in some private JVM thread, the virtual machine may be unusable from that point on.

We take two measures to protect us against this possibility:

1. The script that starts the job must provide an upper bound on the amount of memory that the macro could require. This deals with the problem in the correct way: we can refuse to run jobs that will never be able to complete due to lack of memory, and run multiple jobs on a single server if their memory requirements are low.

2. The job server checkpoints the status of the jobs to disk so that if any catastrophic error occurs, we can exit the VM and restart. If measure 1 is implemented properly this will not be necessary to deal with OutOfMemoryError, but there are other pragmatic reasons why this is important, such as wishing to recover from power failures, allowing the job server to be restarted safely when a new version is available, etc.

Each job server can be configured to take advantage of multiple cores by specifying the maximum number of jobs that may run concurrently. We generally set this to $n + 1$, where n is the number of cores.

3.4.5 Apache-based Front-End

The front-end of the confocal archive is a web interface, served from a variety of CGI scripts. These can be divided into two classes:

1. Ruby CGI scripts invoked directly.

2. Ruby-on-Rails hosted code, invoked via FastCGI.

The basic Ruby CGI scripts are derived from the prototype version of the system, while the Ruby-on-Rails component was written by Seymour Knowles-Barley as part of his MSc project in 2007. Knowles-Barley's application provided excellent Javascript-based facilities for annotating images with landmarks and searching based on the Flybase anatomy ontology, although this lacked the infrastructure and facilities necessary for a complete image archival system such as that I had been developing (e.g. allowing scans to be uploaded by HTTP, on-demand unpacking of images

on the server, etc.). After many changes to the underlying database schema and the applications themselves, I managed to get these two systems to use the same database, including common code for authentication and starting jobs. The "look and feel" of pages from these two systems should now be difficult for the user to tell apart.

From a software development point of view, this combination of the two systems is not as elegant as it might be - there is still some code redundancy between them, and the pure Ruby code has a great deal of "plumbing" which is abstracted away by the Rails framework. Ideally, in the future we will move to an entirely Rails-based front-end, since this will greatly reduce the amount of code on the server side.

3.4.6 Safely Starting Back-end Jobs From Ruby

There is an obvious synchronization issue that we have to be careful of when starting jobs to generate data on one of the back-ends. For example, suppose we have two users who are both choosing to use the Javascript viewer to view the same image stack scaled to 50% with the identical colour-to-channel assignments, an operation which requires a back-end job to unpack the original image into scaled PNGs with the right colour map. A naïve approach may result in two jobs being started, wasting CPU and perhaps overwriting each others' output. We get around this problem by using the atomic property of POSIX directory creation.[27] Essentially, the idea is that the thread of execution that successfully creates the directory for output is the one that must then start the job. The name of a directory must either uniquely determine the output for that job (e.g. by including in some fashion all the parameters required for creating it) or be generated by some random process such that it is vanishingly improbable that another thread will try to create one with the same name.

The sequence of actions taken by the thread that successfully creates the directory then proceeds as follows:

- It creates a new entry in the jobs table to find the database job ID.
- It creates a `.generating` file in the directory which should contain the database job ID.
- If the directory is one that may be pruned by the cleanup dæmon, then it should also create a `.deletable` file in the directory.

[27] I have chosen this mechanism rather than, say, flock or fcntl based locking since we wish to be able to write safely to parts of the filesystem mounted via NFS.

- It makes a TCP connection to port 2061 on one of the back-end servers and asks it to start the job.
- The job ID returned by the job server is written into a column in the database's jobs table.

Any other thread that needs that directory, but discovers that it has already been created, will proceed in the following way: (Note that this is the case when reloading a page after a previous visit to that page caused the job to start.)

- The thread should "touch" (i.e. update the mtime) of a file called `.accessed` in the directory. (The mtime of this file is used by the cleanup dæmon to decide which are the least recently used directories that may be pruned.)
- If the `.completed` file exists, the job completed successfully and the thread can proceed, safely assuming that all the files that it expects to be there will be.
- Otherwise, if there is a `.generating` file then the thread should look up the job in the jobs table based on the database job ID contained in that file. The details in that database entry will tell the thread which host, port and job ID to use to query the status of the job, which it should then report to the user.
- If neither file exists, generation must just have started, so report that to the user.

3.4.7 ImageJ Jobs Currently In Use

Extract Image Properties

This is the plugin which is started to generate the main scan directory on upload. The file is opened using helper methods in util.BatchOpener, written by myself but based heavily on HandleExtraFileTypes by Dr Gregory Jefferis. The main differences from HandleExtraFileTypes which are important for this application are:

- It provides an `open()` method that returns an array of ImagePlus objects, one per channel, without calling `show()` on any of them. (This is important for headless operation.)
- Files are identified as particular types solely by their content (magic numbers, etc.) rather than their file extension. (HandleExtraFileTypes uses a mixture of the two.) We cannot necessarily trust the names of files uploaded to the archive to give us the file type.

- In addition to the array of ImagePlus objects the `open()` method returns a string identifying the loader that was used to open the file. This is our best indication of the type of the file for the rest of the application.

The `Extract_Image_Properties` plugin then writes out the generic image metadata[28] to a file called "`properties-generic`" in YAML format for easy parsing by the Ruby front-end. (The YAML is generated via Ola Bini's JvYAMLb library.) If the file is an LSM file, it then extracts the LSM specific metadata (such as LASER and PMT settings for each channel) to the file "`properties-LSM`", again in YAML format. Finally the plugin extracts thumbnail images (one per channel) so that these may be used in search results.

MIPDriver

This file is from Douglas Cowan's MipJ and is used for two purposes, depending on the parameters passed:

- Generating the "realtime" files necessary for the on-the-fly 3D renderings of MipJ;
- Generating high quality MIP projections, either by raycasting or splatting.

Scale, Merge and Unpack

This plugin is used to generate data for the Javascript-based stack viewer, by unpacking a scan to a series of PNGs. For each slice, there is one PNG per channel, and if the number of channels is greater than 1, there is an additional "merged" image. The allocation of channels to red, green and blue can be specified via macro parameters, although these are ignored for RGB images. The PNG writing code was originally based on the Java Advanced Imaging library's ImageIO class, but for particular cases this appeared to take an extremely long time and the performance of PNG writing is quite important in usability terms with this system. In order to get more predictable and faster performance I replaced the ImageIO calls with a JNI-wrapped library that uses libpng. Depending on the source stack type, this improved performance of PNG writing by between 2x and 4x. (The source code for this package, called "fastpng" is in `VIB/fastpng`.)

[28] i.e. that which is common to every ImagePlus in ImageJ, such as width, height, depth, bit depth, etc.

3.4.8 Cleanup Dæmon

One of the goals for this system is that it should be possible for a group to set it up easily on a standard Linux-based computer, so the server needs to carefully manage the way that it uses disk space. The amount of space taken up by the generated files (e.g. for viewing over the web) can easily be many times that of the original scan. The cleanup dæmon script, started from `/etc/init.d` as any other `sysvinit` service, wakes up every 5 minutes to check:

- The amount of free disk space;
- The amount which could be freed by removing `.deletable` directories.

... and if the amount of space free is less than the configured target amount of free disk space, it removes `.deletable` directories until that amount has been freed up. The directories are removed starting with the least recently used, according to the mtime of the `.accessed` file (if it exists). Any directory that has been created in the past 15 minutes is not considered for removal.

Since directories must be removed as well as created atomically, the directory is removed by first renaming it with a `.moved` extension before removal with `rm -rf`.

3.4.9 Backup Scripts

Backing up the archive is done with a simple script which can be run from `crontab` on a regular basis according to the group's policy. The preliminary steps that it takes are:

- Running `pg_dump` to dump the PostgreSQL database;
- Assembling a list of files to back up by finding all those not contained in `.deletable` directories;
- Using `tar` to archive those files.

These files can then be transferred with `rsync` to off-site backup, written to tape, etc.

3.4.10 Account Creation

Once a new user requests an account on the server the administrators are emailed a notification. There is an administrative interface for administrators to approve account requests and assign the

user to particular groups. This manual step could be removed completely at a later stage if we had much more disk space available; as it is, we are trying to scale up the number of users in a controlled fashion.

3.4.11 Account Management and Security

The new version of the system has a more sophisticated security mechanism than the prototype, allowing users to control the visibility of the scans they have uploaded according to a system of groups. Any user who is marked as an administrator may create new groups, which typically represent research groups, projects or groups of outsiders. Each user is assigned a default group when their account is created; scans uploaded by each user are by default visible to this group. The visibility of each scan (in terms of which groups may view it) can only be changed by the owner (the user that uploaded the file in the first place), who can change this by checking or un-checking boxes on the scan's page. There is a special "Public" group as well - scans made visible to this group can be seen by users without a login.

3.4.12 User Experience[29]

After a user's account has been approved by one of the administrators, they will receive an email with a one-time link that will allow them to set their password. After this, the user may login to the site using the form in the top right of the main screen. It is necessary to be logged-in in order to perform many actions on the website, such as uploading scans, but users may view some of the public scans without authenticating themselves.

In a typical workflow, the first thing that the user might do is to upload one of their own scans. The simplest way to do this is using the HTTP POST based form available from the "upload" link at the top of every page. This presents a form such as that shown in Figure 9, which allows the user to select a filename from their local computer.

After submission it may take some time to upload the image data to the server (and typically web browsers provide very poor feedback during the POSTing of large files). However, afterwards the user will be directed to the scan's main page, as shown in Figure 10. There is a great deal of data

[29] n.b. The reader may wish to follow through these steps at `http://fruitfly.inf.ed.ac.uk/confocal/` where the current public version of the server software is running.

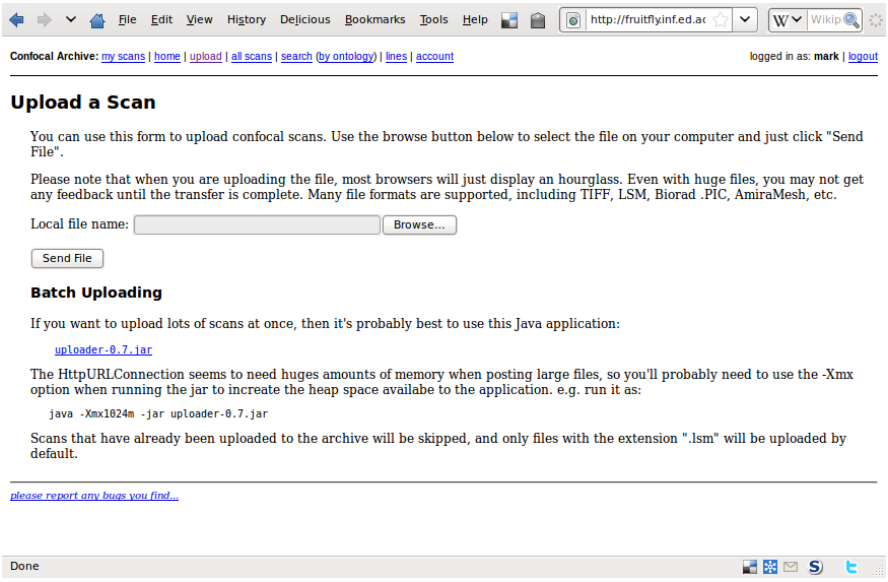


Figure 9 – The "upload" page.

presented on this page, particularly if the uploaded file is a Zeiss LSM file. (For these files the database also extracts data about the PMT settings, objective and LASERs that were used.) The page is divided up in a way that should be clear: the main part of the page shows the immutable data extracted from the scan and various advanced viewing options, while the panel on the right shows any additional annotations added by any user. The most common options for viewing and annotation are shown as links in large text in the top left; these options are "Download" (to download the original file), "View" (for simple stack viewing), "Annotate" (to access the interface for creating landmark tags), "3D Project" (the MipJ Java applet for creating 3D projections) and "Delete" (only available if the user owns the scan, and even then the action requires confirmation).

The "View" screen is shown in Figure 11. The controls at the top of the page allow one to move about in the stack, choose to zoom out or select which channel to display. (The slider control is free software from the Yahoo User Interface library.[30])

The "Annotate" screen is shown in Figure 12. This allows the user to create a landmark tag attached to a particular point in the stack, indicated by the circle-and-crosshair on the image. The landmark tag's annotation is free text but with auto-completion of names in the Flybase anatomy ontology. (Auto-completed phrases are kept in Wiki-style square brackets for easier searching by ontology

[30] `http://developer.yahoo.com/yui/`

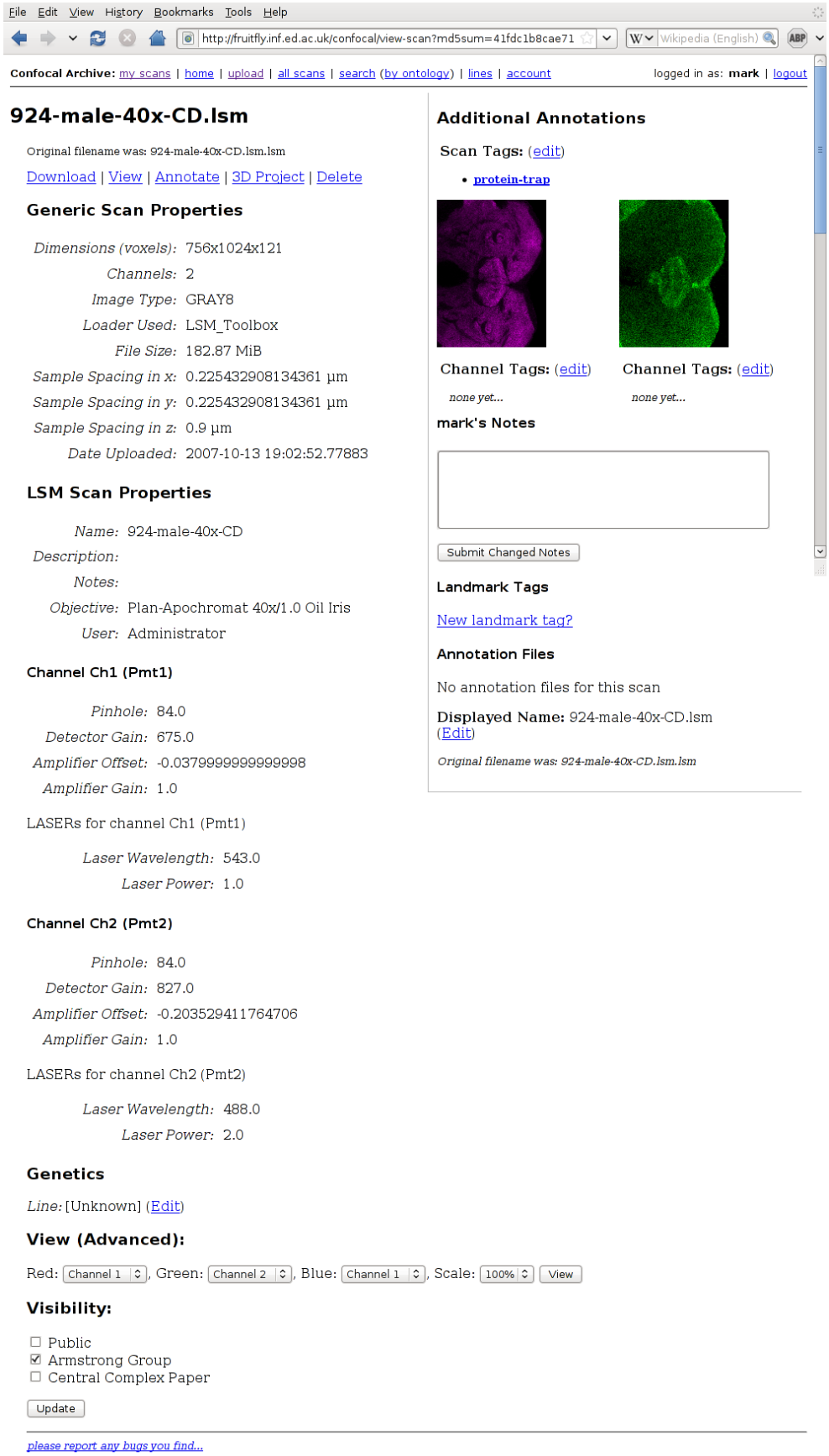


Figure 10 – The "view scan" page.

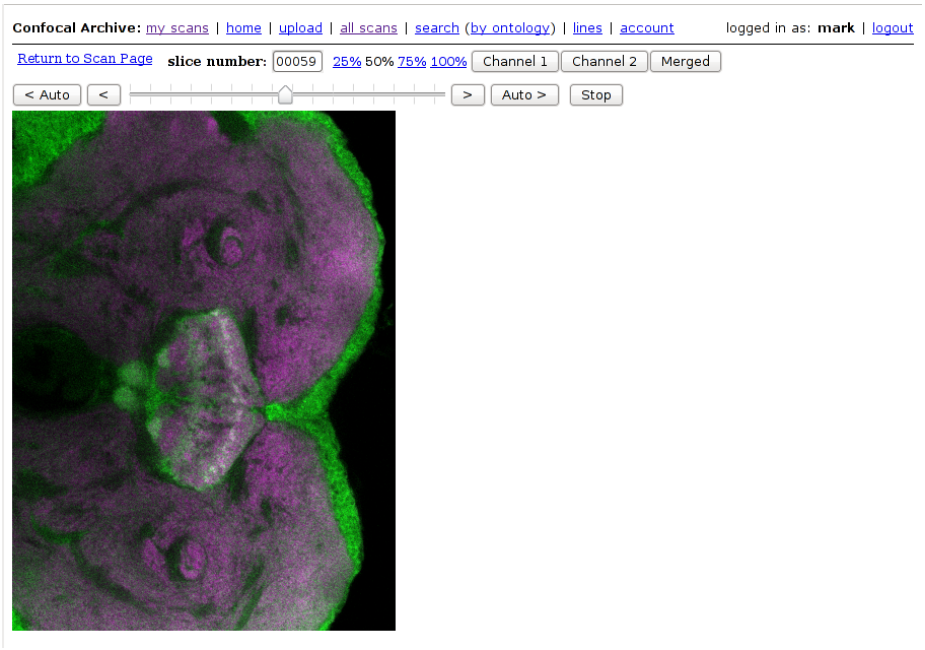


Figure 11 – The Javascript viewer (via the "View" link).

terms.) This interface is further described in *[Knowles-Barley, 2007]*. The Flybase anatomy ontology provides a controlled vocabulary to describe regions of the fly brain with semantic relationships defined between the terms, such as "is_a", "develops_from" and "part_of". Labelling parts of the brain with these terms in this semi-structured fashion allows simple and unambiguous searching of the scans for particular features. In the future we might hope to be able to infer such annotations automatically, but for the moment annotation systems such as this are the best way to add semantic mark-up to fly brain images.

The MipJ viewer page is shown in Figure 13. The user can select the different channels via the drop-down "Channels" menu, and rotate the stack by clicking and dragging in the image. This fast real-time view has some viewing artefacts, but to generate a high quality projection the user should click "Generate Projection" once they are happy with the selected view. The original development of this applet-based renderer is discussed in [Cowan, 2003].

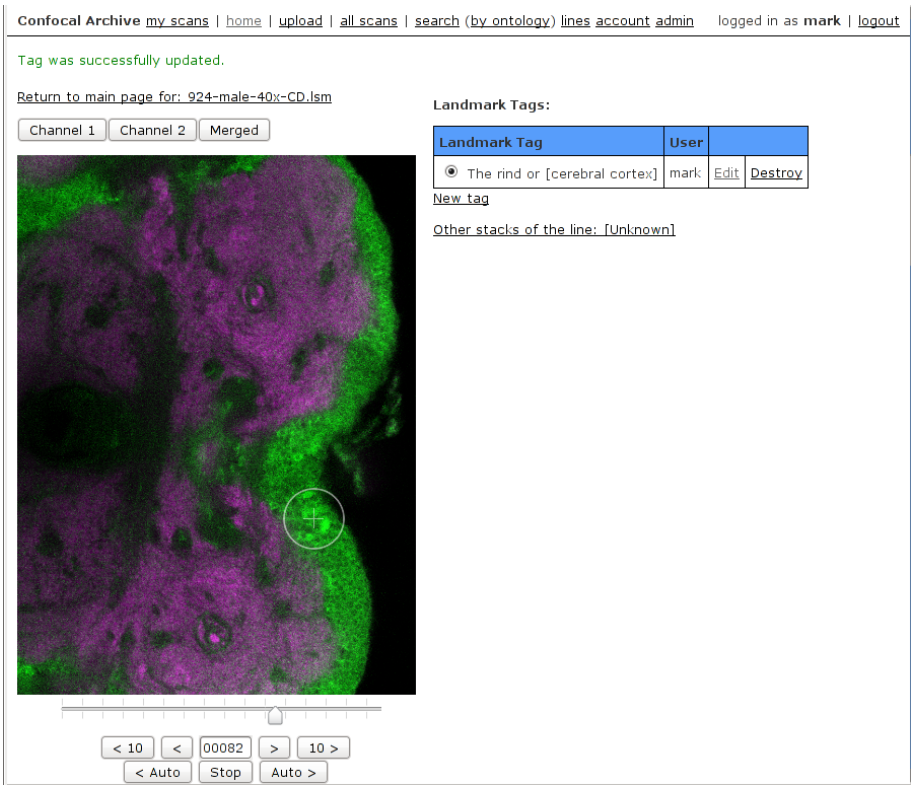


Figure 12 – The landmark tag interface (via the "Annotate" link).

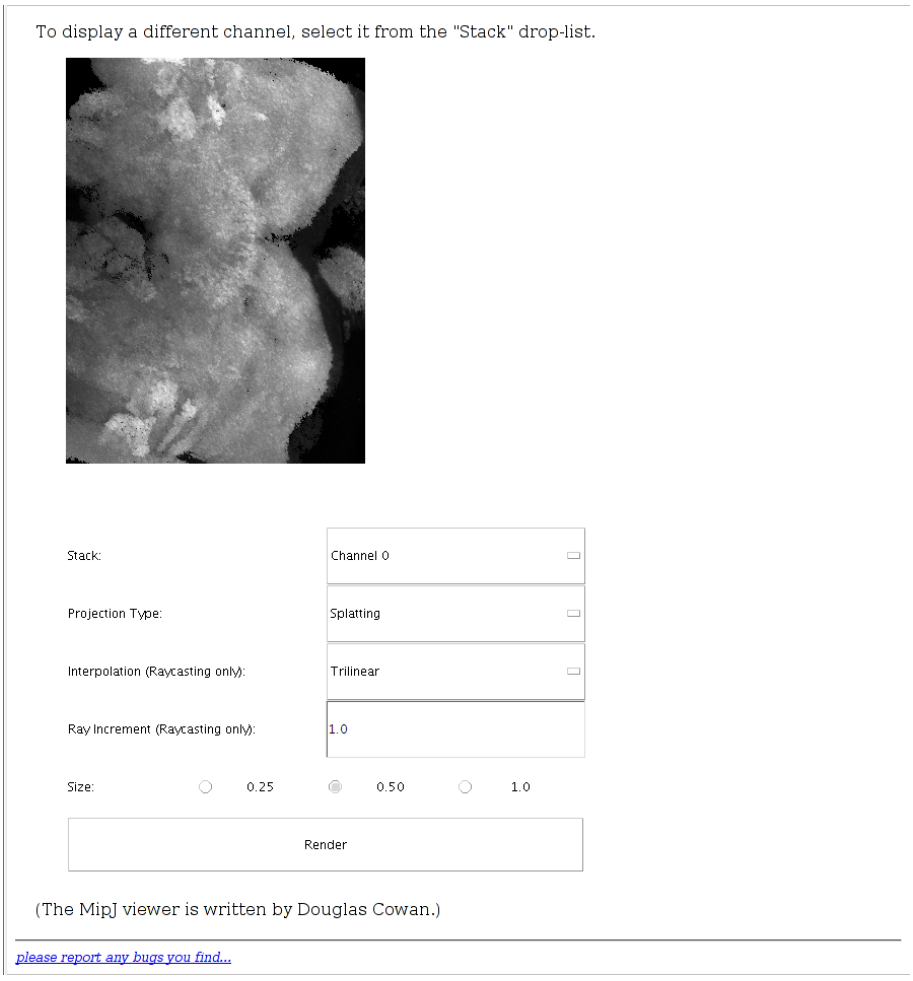


Figure 13 – The MipJ viewer Java applet (via the "3D Project" link).

3.5 Evaluation

The evaluation of software such as this, which is strongly focussed on the user experience, is via feedback from test groups of users. This essentially took place in three stages: informal beta testing and then two stages of more organized feedback requests, from local and remote users.

3.5.1 Beta Testing

I was given very useful feedback on the initial versions of this software from Dr Bilal Malik, Seymour Knowles-Barley and Dr Douglas Armstrong. In particular, they raised the following issues, which I have paraphrased here with their resolutions:

- "Waiting for PNG generation is sometimes frustrating, and the page that shows the job progress should automatically refresh."
  - I changed the holding page to refresh every 5 seconds.
  - I switched the PNG generation from ImageIO-based code to JNI-wrapped use of libpng.
- "The default view of the scans should show the merged view rather than the first channel."
  - I made this change.
- "Display of the 'all scans' page is very slow."
  - This turned out to be a bug in the SQL statement used to return the results, which I fixed.
- "It should be possible to change the displayed name of the scan."
  - I changed the system so that this is possible, although the original filename is still shown as well.
- "On upload, the PNG generation for the 'view' and 'annotate' pages should be started immediately."
  - Due to the design of the system this is actually difficult to arrange. This is discussed further under "Job Server Interface" in section 3.6.5.

Naturally, there were many other small bugs and problems that were found during this beta testing, but the above were the most frequent complaints.

3.5.2 First Round (Local Users)

I asked various users in Edinburgh fly groups, including the Armstrong, Jarman and Pennetta groups, to try the following tasks on the database:

- Go to `http://fruitfly.inf.ed.ac.uk/confocal/` and create an account. (You will receive an email with further instructions after your account creation request has been approved.)
- Once you have followed the account creation steps, login to the website at the URL given in the previous step using your user name and password.
- Try uploading a multi-channel confocal image stack via the "upload" link on the web page. (We suggest using an LSM or TIFF file, although many other formats should work.)
- View the scan using the "view" link. Move through the stack to see different slices, and try different channels.
- Try creating a landmark tag at a particular point in the stack by using the "annotate" link from the scan's page.
- Try generating a 3D projection of one of the scan's channels by following the "3D Project" link.
- Use the "all scans" link to browse all the uploaded scans that are visible to you.
- Please rate on a scale of 1 (very difficult) to 5 (very easy) how easy it was to perform these steps.
- Please describe any problems you had with the steps above.
- What facilities would you like to see on a website such as this? (Various image processing tasks could be performed automatically on the server, for example.)

Of the 21 people asked for feedback in this round, only 2 answered the most easily scoreable question, rating it as 4.5 and 5 out of 5 respectively for ease of use. However, more general comments were received from about 5 respondents. These were very positive, but raised the following potential problems:

- It was not clear to two users how to return to the previous page in some places, so I added prominent links for this in the situations they mentioned.
- It was not obvious to one user that you could click and drag in the MipJ applet to rotate the image in 3D, so I added some explanatory notes at the top of that page.
- One user requested more complex annotation possibilities than the landmark tags, such as being able to paint over particular regions or draw arrows. While the former is a good idea, it would be time-consuming to implement, so I decided to leave that for further work. The latter seemed to me to be a cosmetic difference from the current system, so I decided not to make that change either.
- Someone requested larger views than 100%, so I added options for 200%, 300% and 400%.
- One user complained that the Javascript viewer went through the stack too quickly, so I added an option to set the number of frames-per-second or allow the viewer to scroll as fast as possible.
- Various small cosmetic issues were mentioned by several users and quickly fixed.
- One user requested a simple maximum intensity projection in Z as a separate option from the MIP projection, which essentially does the same from multiple angles. I have added this to my list of requested features.
- One user asked about the access controls - currently only users whose accounts are marked as "administrators" can set up new groups, and change the default visibility of each user's newly uploaded scans. I have plans to make the permissions system more flexible, but this is not yet implemented.

3.5.3 Second Round (Remote Users)

Once the issues discovered via the first round of testing were resolved, I sent the same feedback request to six ImageJ and Fiji developers working in remote institutions. The reason for requesting this feedback in a second round was that remote users are likely to find the Javascript stack viewers and various other parts of the site less responsive, and I wanted to address any more general usability issues with the first round of testing. The responses were again very positive, with just the following bugs or requests made by the testers:

- One tester suggested using JPEG encoded images in the Javascript viewer. I had resisted using JPEG since as a lossy encoding I felt it was inappropriate for biological image analysis. However, since using this encoding enormously improves the responsiveness of the Javascript viewer, I added it as an option.
- The security model was unclear to one of the testers, to whom it wasn't obvious that certain scans were deliberately made public. Despite this, I thought that the distinction was sufficiently clear already, since when not logged in there is a warning at the top of the list of scans indicating that only the few public scans are visible.
- At the suggestion of one tester, I removed the "Clear Form" buttons thoughout the site - they were said to be a nuisance.
- I made the image thumbnails on the "View Scan" page clickable in response to a request from one of the testers.

3.6 Future Work

The system as it stands works well, but there are many possible ways in which this work could be taken forward if there is seen to be sufficient enthusiasm for them. The following sections discuss some of these possibilities.

3.6.1 Server Side Image Processing

The use of ImageJ macros for server side image processing suggests a huge range of possibilities, in particular:

- Automatic cell-counting, such as is implemented by OME server's FindSpots functionality.
- A "best effort" registration of images to one of a number of standard templates on the server.
- Automatic neuron tracing.
- Multi-scale Gaussian convolutions of the images: the first step in many advanced image processing algorithms is to calculate Gaussian convolutions of the image with various kernel sizes, but this can be an enormously time-consuming process. However, by pushing this onto a

fast server (possibly with custom hardware or mass-market GPUs) this CPU-intensive task could be done once and cached remotely.

3.6.2 Annotation Types

It might be argued that as the system is currently implemented there are too many different types of annotation, and that this will be confusing to users. However, my view of this is that the annotation system is there to be used as each user wishes for their data set. For example, some may prefer the "folksonomy" style of adding single-word "tags" to scans, whereas some may prefer to use free text notes. Similarly, the landmark tags (associated with a particular point in the stack) are undoubtedly conceptually different from the other types. By way of comparison, many "Web 2.0" websites such as Flickr have a similar breadth of annotation types without apparently confusing the user.

The clarity of the interface to these different annotations could certainly be improved in a number of ways, however. To take two small examples: (a) the auto-completion of ontology terms in annotations is only available for landmark tags at the moment, and (b) while it is possible to use landmark tags as a method of leaving comments on other people's scans, this is not presented in a way which seems natural compared to the comments systems on other web sites. Another key issue is that the terminology used at the moment seems to be confusing to some people, and in the future we should rationalize this.

In the future, the system may also benefit from having some semantic metadata attached to elements of the scans' pages, for example to mark the genetic line in a machine-readable fashion. Semantic metadata on the world-wide web, typically in the form of microformats or RDFa, has been slow to take off, but would be an ideal way to enable searching across multiple instances of the software run by different organizations.

3.6.3 Installable Packages

At the moment, installing the front-end and back-end of the server is a complex process, which means that it would be difficult for other groups to try it out on their own local networks. Biological research groups are naturally wary of uploading their data to external services, so the vision for this project has always been that each group should be able to set up the system for themselves on a private network. In order to make this viable, it is important that the installation is as simple

as installing one or two packages on a Debian or Ubuntu based server. I have made some progress towards this goal, for example by producing Debian packages of Fiji, and hope to be able to complete the packaging soon.

3.6.4 Code Simplification

The split between the Rails and non-Rails parts of the front-end is inelegant, and the sections of the code which are not based on the Rails framework are more verbose and consequently more error-prone and difficult to maintain. I am planning with Seymour Knowles-Barley to rewrite these remaining sections using Rails.

3.6.5 Multi-File Based Formats

Perhaps the most important limitation of the current design is that it only works well with single-file based formats, such as LSM, TIFF, etc. In contrast, sometimes one has to deal with formats that are made up of directories of files, one file per channel (like the BioRad .PIC format), or one file with a separate metadata file (such as NRRD). The two problems with these formats are:

- The user can only upload a single file with HTTP POST and the <INPUT TYPE="FILE" ...> HTML form element. Uploading multi-file formats would have to be done in the uploader application by first packing the files in some predictable way.
- The MD5sum of a packed set of directories is less easy to check, and using this to avoid duplicates would mean having to carefully pick a packed format such that no matter what the glob order, mtimes, etc. of the files are, they will still produce the same archive.

Neither of these is by any means an insoluble problem. A careful application of `tar` in the uploader would suffice, or it may be simpler to use an off-the-shelf uploader widget. However, up to this point I have not had time to implement any of these options.

Job Server Interface The interface to the ImageJ-based job server is not very sophisticated at the moment, and this could create certain problems:

- Calculation of the amount of memory that each command will require is currently done in the Ruby code; it would be more elegant (and simpler to maintain) to include this logic in the job server.
- There is currently no mechanism for the job server to chain dependent jobs together. This is an important point because one of the most frequently requested features of the website is to pre-generate the PNG files for the default view of a scan immediately on upload. This can only take place once the job to extract the basic image properties has completed. In this case, the simplest solution would be for the former job to enqueue the latter once it has completed, which is currently not possible.

3.6.6 Usability and Presentation

There are a large number of ways in which the usability of the site could be improved, and it is likely that these would drive use of it more than adding new features. For example:

- The uploader application is somewhat awkward to use, which discourages people from uploading images in bulk in the first place.
- It is not obvious to users how to create "albums" of scans via tagging, although this is quite possible.
- Various operations which could be conducted via AJAX operations currently require an extra two page loads.

Another frequently requested feature is to be able to add a custom front-end for particular sets of scans, for instance to make the publication of some set of scans for a paper appear as a standalone site.

3.7 Conclusions

While I consider this to be a successful project so far, the future of it needs some careful consideration. Perhaps the biggest question is whether it is worth continuing to develop this database as a standalone project. Other options include splitting the best functionality into separate smaller projects or trying to incorporate them into OME Server. Alternatively, developing an interface to exchange data

with OME Server may be a useful safeguard for the future. In any case, I will shortly merge job server back-end into Fiji soon, since this component may be more generally useful to the ImageJ development community for building similar systems. Once easily installable packages of the complete system are finished and we have made a concerted effort to publicize the project, we can be better guided in how to proceed by feedback from a larger group of users.

4 Image Registration

4.1 Background

In any application of biological or medical imaging in which it is necessary to aggregate or compare data from multiple images (whether they were acquired in a time series, from multiple subjects, etc.) the requirement for high quality image registration inevitably arises. This is the process that allows us to identify a sample at a point in one image with one in another such that we can meaningfully relate the two images.

The sense of the verb "register" meaning "to put into alignment" was actually used as early as 1839 in the printing industry, [OED, 1998] where it was used to describe the process of overlaying two impressions such that they fit as well as possible. Pleasingly, this is still an apt metaphor for the way that the word is used in modern image processing - the problem is still to overlay two images such that they line up. However, when the images are from modern light microscopes, the problem is made vastly more complex, since the images are 3D image stacks and may require some stretching and distortion in order to make a good match.

The object of this chapter is to consider a number of candidate registration techniques and evaluate how well they perform on the image corpus I have collected. This is important because there is no general purpose registration tool that performs consistently across every data set; in this area the techniques applied tend to be strongly tailored to the properties of the data being studied. Even with particular modalities, comparisons of the performance of large numbers of registration techniques are rarely done [Holden, 2008]. If our images were of whole brains with consistently high-quality neuropil staining, we would most likely appeal to previous work (e.g. [Jenett et al., 2006] or [Jefferis et al., 2007]) and pick a method. However, the nc82 staining in this corpus is rather variable (with large amounts of noise in some scans) and we have only a subpart of the central brain to consider. This raises the possibilities that standard approaches may fail and that simpler techniques may work well enough. I also take care in this chapter to attempt to establish that our automatic results do bear some correspondence to expert human evaluations of the registrations: a consideration that is often overlooked.

In section 4.3 we will review the use of registration techniques for insect neuroanatomy, but first it is worth making a few points about the fundamental difficulties of assessing these methods.

4.2 Difficulties of Image Registration

4.2.1 Ill-defined Problem

In most situations where we attempt to register two images of different subjects (e.g. the brains of two different flies) it is important to be clear that there is no "ground truth", in that there is no guarantee that any definable part of one brain exists in another. Even if one can identify one neuropil region with a similarly shaped one in another image there is no way of establishing the correct mapping between any two points within that region. It is not even clear what "correct" would mean in that situation: developmentally similar neurons may end up in different positions within a neuropil region in different flies due to neural plasticity.

As a result, it is unclear how we define quality of registration, either for evaluation purposes or for the optimization stage of many registration algorithms. This will be discussed further in section 4.7, where we will consider a number of measures and evaluate them according to how well they correlate with the judgement of human experts.

4.2.2 Sources of Error

It is instructive to consider the possible sources of error when using registration to identify points in two different fly brains. At the very least, these include:

- Morphological differences due to developmental variation
- Dissection damage
- Shrinkage during fixation and preparation
- Poor signal from fluorescent reporters
- Drop-off of signal in the Z direction
- Excessive noise in the Z direction
- Movement of the confocal microscope during acquisition
- Registration error

The tendency is to focus on the last of these since it is the only step which may be repeated (or repeated without losing information), and is the only stage where algorithmic improvements are generally implementable. However, when we are depending on being able to distinguish structures that may be separated by 1 or 2 μm (such as neurons in the fan-shaped body) errors in the earlier stages may be very difficult to compensate for at the registration stage.

4.3 Review of Registration Techniques

In this section I will briefly review techniques for 3D image registration. Firstly, we should clarify some terminology:

- The "template image" is the image onto which we wish to map all of the others.
- The "model image" is one of the images which we wish to map onto the template.
- In this chapter I use the term "registration" rather broadly. For example, some people would consider finding correspondences between landmarks in two images to be "registration" and the later step of finding a mapping of all other points in the image based on the landmarks to be "transformation". In this chapter I use "registration" to cover both of these steps, as well as any earlier steps required to find the landmarks in the first place.
- I use "elastic" synonymously with "non-linear" when discussing transformations, although various authors use this term in a more specific sense.

A convenient way of dividing up registration algorithms is according to the type of transformations they can produce, the most obvious such categorization being between linear and non-linear transformations.

4.3.1 Linear registration

Linear transformations have the property that they always map lines in one image onto lines in the other. In 3D a general linear mapping from $\mathbb{R}^3 \rightarrow \mathbb{R}^3$ can be defined by a homogenous matrix M:

$$M = \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

which can map a point $p = (p_x, p_y, p_z, 1)$ to a point $q = (q_x, q_y, q_z, 1)$ by matrix multiplication:

$$\begin{pmatrix} q_x \\ q_y \\ q_z \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} p_x \\ p_y \\ p_z \\ 1 \end{pmatrix}$$

The simplest linear transformations used in image registration are rigid mappings, which are defined by 3 parameters for rotation and 3 for translation, so if the rotation is expressed in Euler angles α, β and γ and the translation is (t_x, t_y, t_z) then M would be:

$$M = \begin{pmatrix} cos(\alpha)cos(\gamma) - cos(\beta)sin(\alpha)cos(\gamma) & -sin(\gamma)cos(\alpha) - cos(\beta)sin(\alpha)cos(\gamma) & sin(\beta)sin(\alpha) & t_x \\ cos(\gamma)sin(\alpha) + cos(\beta)cos(\alpha)sin(\gamma) & -sin(\gamma)sin(\alpha) + cos(\beta)cos(\alpha)cos(\gamma) & -sin(\beta)cos(\alpha) & t_y \\ sin(\beta)sin(\gamma) & sin(\beta)cos(\gamma) & cos(\beta) & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

If we allow scaling as well, the next class of transformation is "rigid + scaling", sometimes known as a 9 degree of freedom affine transformation. This is composed by post-multiplying the rigid M above with a pure scaling matrix:

$$\begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

A full affine transformation (with 12 degrees of freedom) also includes three shearing parameters. (In this chapter we use "affine" to mean a 12 degree of freedom affine transformation (i.e. one including shearing) and "rigid + scaling" to refer to a 9 degree of freedom affine transformation.)

Linear transformations are mostly used in the following situations nowadays:

- Registrations of the same subject where no morphological changes are expected between images.
- As a first transformation from which an elastic mapping is elaborated.
- Registrations of small regions of brains, which might be combined with other such local registrations.
- In the case of rigid transformations, when one wants to compare the volumes of registered regions.

Linear registrations are typically found using the following broadly described schema:

1. Some measure of similarity between images is defined (e.g. mean of squared difference, mutual information). This may include a penalty for excessive shearing or rotation away from some approximate alignment.
2. The parameters for an initial transformation are guessed, and a standard optimization algorithm[31] is used to optimize the similarity measure with respect to the transformation's parameters.
3. The previous step may be repeated for a number of different initial guesses.

A huge number of registration algorithms can be fitted into this schema. In some other cases (e.g. rigid transformations from landmarks, with similarity defined by squared distances) it is possible to calculate the optimum transformation directly. This particular case is discussed below in section 4.4.1. Another example of a technique which avoids the iterative optimization stage for linear optimization is to use the Fourier Shift Theorem to effectively try all possible translations of images at once [Preibisch, 2008].

4.3.2 Non-linear registration

Non-linear registration is an active field of research and many innovative (and often complex) approaches to the problem have been suggested. This section will necessarily provide only a

[31] Numerical optimization is an involved field, so I do not describe the options for these algorithms here - [Nocedal and Wright, 2006] is a comprehensive reference for these.

superficial summary of some of these, but more detailed surveys can be found in [Toga, 1999] and [Holden, 2008].

The transformation produced by a non-rigid registration algorithm may be a vector field giving displacements for each point in the image. It may also be described in terms of some smaller number of parameters, such as with spline based methods. Splines are simply functions on some one-dimensional interval which are piecewise polynomial between points known as knots. (Where the knots in the interval are equally spaced, the spline is referred to as uniform.) The domain of the spline may be multi-dimensional. The degree of a spline is said to be the maximum degree of any of the polynomials. Splines provide a convenient way of defining curves and flexible interpolations between points, while avoiding Runge's phenomenon that creates problems when fitting a single polynomial.[32]

B-splines (short for "basis splines") can be seen as a generalization of Bezier curves. For each knot in the spline there is a control point, and the value of the B-spline at a point t is made up of the control points weighted with the value of a series of basis functions in t. In particular, these are used in [Rueckert et al., 1999] to define a free-form deformation based on the movement of a 3D grid of these control points, which is optimized for normalized mutual information.

An alternative way of describing deformations of images in terms of basis functions is to use radial basis functions, as suggested in [Fornefett et al., 2001] - these are functions based on distance of points from given landmarks. A frequently used radial basis function technique is that of thin plate splines. Mathematically, thin plate splines were originally used to describe deformations of metal sheets, but they are also commonly used to deform images such that landmark points match up [Bookstein, 1989]. The thin plate spline technique finds the minimum energy solution where the landmarks are put into correspondence, but it has an important drawback in that the effect of adjusting a single point affects the whole transformation, so single misplaced points can create large and unwanted distortions. In contrast, adjusting a control point in a free-form deformation based on cubic B-splines only affects the region around that point.

Physically Inspired Registration Techniques: Another class of registration technique (although less used in practice for biomedical image registration) consists of those inspired by the physics of deformations in the real world. Fluid registration techniques, for example, can model

[32] `http://en.wikipedia.org/wiki/Runge's_phenomenon` on 2009-09-10

large deformations easily as the flow of a viscous fluid [Christensen et al., 1994]. A similar idea is to model the deformations in images as optical flow [Thirion, 1998]. An alternative physical model of deformations are "elastic body splines" which describe deformation of an image in terms of forces applying to an elastic body [Davis et al., 1997, Kohlrausch et al., 2005].

4.3.3 Registration in Insect Neuroanatomy

There have been a number of projects to generate standard template brains or atlases of model insect organisms, and the registration techniques used in some of these are briefly described below:

The fruit fly *(Drosophila melanogaster):* The *Drosophila* Standard Brain [Rein et al., 2002] was generated by registering confocal images of fly brains with a variety of registration techniques implemented in the Virtual Insect Brain (VIB) protocol. The recommended VIB method is the diffusion interpolation method described below in section 4.4.4.

The honey bee *(Apis mellifera):* The process of constructing the standard atlas of the honey bee brain is described in [Brandt et al., 2005]. The registration method used is that described in [Rohlfing et al., 2001] and based on the cubic B-spline method in [Rueckert et al., 1999]. This registration technique was later implemented in the Computational Morphometry Toolkit described below in section 4.4.5.

The desert locust *(Schistocerca gregaria):* [Kurylas et al., 2008] describes the process of generating the standard locust brain atlas. This used two registration methods: firstly, the VIB protocol, as used in the *Drosophila* standard brain, and secondly the [Rohlfing and C. R. Maurer, 2003] implementation of the algorithm in [Rueckert et al., 1999] - this latter approach builds on the honey bee technique mentioned above.

4.4 Registration Methods Considered Here

There are a very large number of techniques for image registration that have been developed and described in publications, and in order to articulate why the particular techniques chosen in this study were picked it is useful to state our basic requirements:

1. We will not consider any techniques or tools that do not work on 3D image stacks that contain brains in arbitrary orientations. This requirement excludes a large number of ImageJ tools that only currently register 2D images, e.g. bUnwarpJ, SIFT landmark-based registration, etc.

2. There should be an accessible existing implementation, or the method should be simple to implement. The point of the survey in this chapter is not to create new registration techniques nor to implement theoretical methods, but instead to find which existing techniques work well for this particular problem, i.e. registering the data collected as described in Chapter 2.7.

3. The registration technique should work on part-brain images. Some techniques, such as PCA-based alignment, only work well for aligning whole brains.

For the techniques that meet these requirements, we chiefly care about the following properties:

- Based on which data does the technique calculate a registration? Some of the techniques we consider start from sets of landmark points, some start from labelled volumes and some use the raw grey data of the image stack. (This is a slightly fuzzy categorization since a tool may use automated landmark extraction followed by a registration technique that only considers the landmark points; similarly some tools that operate on grey values work better with an initial guessed registration based on manually chosen landmarks.)

- How much manual mark-up is required in order to use the method? In other words, how time-consuming is it to annotate the brain for the registration technique?

- How long does the automated part of the method take to produce output? (i.e. how computationally expensive is it?)

- On what software platform does the tool work, and under what terms is it licensed? We strongly prefer techniques which are free software and those that run on the ImageJ platform.

- Does the tool have a published record of being successfully used for registration of *Drosophila* neuropil images?

The following table lists the techniques considered in the rest of the chapter, and summarizes some of these properties:

Technique or Tool	Output Transformation	Input Data	Markup Time	Run Time	Platform	License
Rigid + Scaling (**rigid**)	Rigid + Scaling	Landmark Points	~ 1 min	~ 1 min	ImageJ	GPL
Affine (**affine** and **affineo**)	Full Affine	Landmark Points	~ 1 min	~ 1 min	ImageJ	GPL
Thin Plate Spline (**tps**)	Spline Transformation	Landmark Points	~ 1 min	~ 5 min	ImageJ	GPL
VIB (ImageJ) (**vib**)	Vector Field	Label Fields	~30-40 min	~ 40 min	ImageJ	GPL
CMTK[33](**cmtk**)	Vector Field	Grey Values[34]	~ 1 min[35]	~ 1 hour	Windows / Linux / MacOS	GPL

4.4.1 Rigid + Scaling

This method (**rigid**) generates transformations composed of rotation, translation and scaling. (I excluded pure rigid transformations (i.e. those solely consisting of rotations and translations since it was established early on that there is a significant difference in the size of the brains found in the image corpus.) The transformation calculated is the best transformation of landmarks based on least squares, computed using Horn's solution in quaternions [Horn, 1987]. I used Dr Johannes Schindelin's implementation of this method in the VIB repository.

It may seem odd that we are even considering non-elastic registration methods such as this in the current survey, since the accepted wisdom is that elastic methods are required for good registrations of fly brains. However, we are considering only quite a small region of the central brain which excludes the typically problematic optic lobes, so it is worth examining whether this assumption still applies for this set of images.

4.4.2 Affine

Affine registrations are distinct from the previous method in that they also allow for shearing in three axes - they are the most general type of linear mapping. Unlike rigid + scaling transformations, we have to find affine transformations by numerical optimization. Two separate implementations of this are considered here:

- **affine**: The transformations defined by every possible combination of four landmarks from the complete set are considered, and the best of these (in a least squares sense) is chosen.

[33]The Computational Morphometry Toolkit by Torsten Rohlfing and Calvin Maurer, the new name for the tool described in [Rohlfing and C. R. Maurer, 2003]. These tools are sometimes referred to as the Flybrain @ Stanford tools.

[34]Optionally one may use landmark points to suggest the initial affine transformation.

[35]This assumes that one suggests initial landmarks.

- **affineo** (for "affine optimized"): The transformation is optimized using GNU R's implementation of BFGS (see Chapter 6 of [Nocedal and Wright, 2006]) from an initial guess. These transformations are calculated using code written by Dr Gregory Jefferis, and included in the `Affine.R` code in the Flybrain @ Stanford tools.[36]

It may be unclear why the former method is included, since one would expect the latter to work better in general. Indeed, had this been the case I would have omitted it entirely, but it became apparent that when scored based on grey values (as opposed to just the mapping of the landmarks) there are a number of cases where the former method does better. Ideally both of these methods would be replaced by a combined approach, where the BFGS optimization is attempted beginning from each start point calculated by the first approach; this would always find the better of the two based on landmarks, but of course would not guarantee that the transformation will be better when scored on grey values.

4.4.3 Thin Plate Spline

The thin plate spline mapping based on landmarks is a smooth non-linear function that maps one set of points onto another with the minimum bending energy. The chief advantages of this mapping for use in elastic transformations of brain values are that it is a C^∞ smooth function, there is a closed form solution and with small numbers of landmarks the results appear biologically plausible. In this survey I have used Johannes Schindelin's implementation in the VIB repository. More details about the derivation of this solution can be found in Fred L. Bookstein's chapter of "Brain Warping" [Toga, 1999].

A common objection to the use of thin plate spline is that it is slow to calculate, but with the small number of manually chosen landmarks we are using performance is not an issue. There is a more complex variation of the mapping known as "smoothing thin plate spline", in which a parameter allows one to specify how non-rigid the transformation may be, but in this survey we have only considered the basic version, in which points are required to match exactly. In practice, this did not seem to create a problem.

[36] These tools can be found at `http://flybrain.stanford.edu/` and are supplemental to [Jefferis et al., 2007].

4.4.4 The Virtual Insect Brain Protocol

The Virtual Insect Brain (VIB) protocol was developed by Prof Martin Heisenberg's group in Würzburg for registration of whole brain images of *Drosophila melanogaster*. An early version is described in the seminal paper on the *Drosophila* Standard Brain [Rein et al., 2002] and a further-developed Amira version is described in [Jenett et al., 2006]. The VIB protocol has options for a number of different registration techniques, but we have chosen to focus on that referred to as `VIBdiffusionTransformation`, which the latter paper describes as the most precise. As a broad description, this method relies on having manually labelled particular neuropil regions of the *Drosophila* brain: rigid mappings are found for each of these neuropil regions onto the corresponding labelled region in a template brain and an elastic mapping for points outside these regions of the brain is found by interpolating these rigid mappings according to the heat equation.

[Jenett et al., 2006] describes the implementation of the VIB protocol based on Amira, which is non-free and is the subject of frequent complaints about poor developer support; as a result we have chosen to evaluate a recent implementation by Dr Johannes Schindelin and Benjamin Schmid which is based on ImageJ rather than Amira, and for which the source code is freely available. Since this reimplementation is not published elsewhere, I will briefly describe its components below.

The actions of the protocol are divided into modules, each of which is implemented as a class that extends the abstract class `Module`. These are described briefly below, and the dependencies of the modules are shown in Figure 14. The three alternatives for `TransformImages` reflect the different options for eventual registration. In this case we are using `LabelDiffusionTransformation`, which is analogous to `VIBdiffusionTransformation` in the Amira version.

1. `EndModule` - this is the top level module which indirectly calls all the others.
2. `AverageBrain` - this uses the `AverageBrain_` plugin to generate averaged images of the warped images for each channel and averages the warped label files to show their relative overlap.
3. `Show` - this displays the images generated by `AverageBrain`, setting a "heatmap" colour map for the averaged labels file.
4. `TransformImages` - depending on the transformation method selected, this runs one of the `GreyTransformation`, `CenterTransformation` or `LabelDiffusionTransformation` modules.

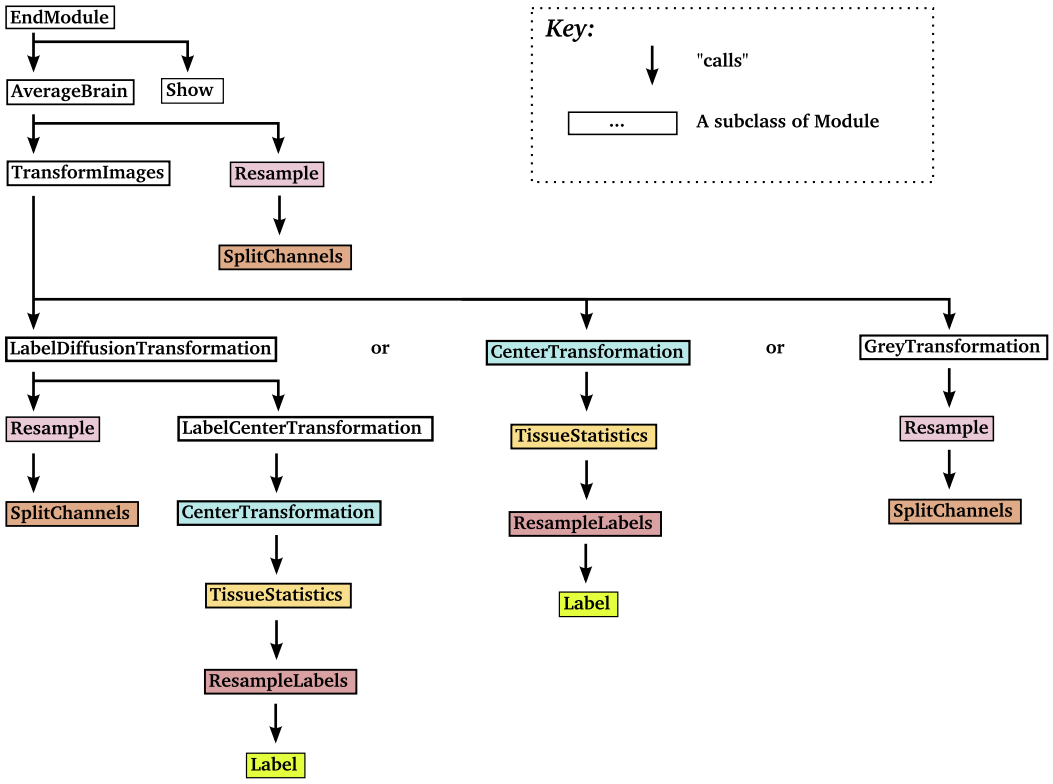


Figure 14 – The relationship between subclasses of `Module` in the ImageJ implementation of the VIB protocol.

5. `Label` - if there is no label file corresponding to the image this loads the `Segmentation_Editor` plugin for manual labelling of the neuropil regions.

6. `GreyTransformation` - this just does a rigid registration on the grey values of the reference channel and stores the transformation in the state.

7. `CenterTransformation` - this finds the best rigid transformation based on the locations of the centroids of each neuropil region.

8. `SplitChannels` - this module splits multi-channel images into separate files.

9. `LabelCenterTransformation` - this module initially works out a simple rigid registration of the image using `CenterTransformation`. Then a rigid registration is found for each individual neuropil region using that overall rigid registration as a starting guess. Each of these registrations is calculated by the `RigidRestration_` plugin's static `rigidRegistration` method.

10. `Resample` - after calling `SplitChannels` this uses the `Resample_` plugin to sample down the image by the selected factor.

11. `ResampleLabels` - this module ensures that the images are labelled (using `Label`) and resamples the label files by the selected factor.

12. `TissueStatistics` - This uses the `TissueStatistics_` plugin to calculate properties of the different labelled neuropil regions. This includes the centroid of the region, the bounding box and the volume of the region. These are then stored in the `statistics` subdirectory.

13. `LabelDiffusionTransformation` - this module uses `LabelCenterTransformation` to generate registrations of each neuropil region and then uses the `DiffusionInterpol2_` plugin to diffuse these transformations throughout the whole image space.

The time-consuming stage of the VIB protocol is the manual labelling of neuropil regions. This was done using the `Segmentation_Editor` plugin, distributed with the ImageJ VIB protocol and bundled in Fiji. A screenshot is shown in Figure 15.

In each brain I labelled the fan-shaped body, ellipsoid body, protocerebral bridge and noduli. (In a few cases the noduli were effectively impossible to pick out, in which case they were omitted.) To speed up this process, the segmentation was done with a simple Wacom pen tablet. The annotation

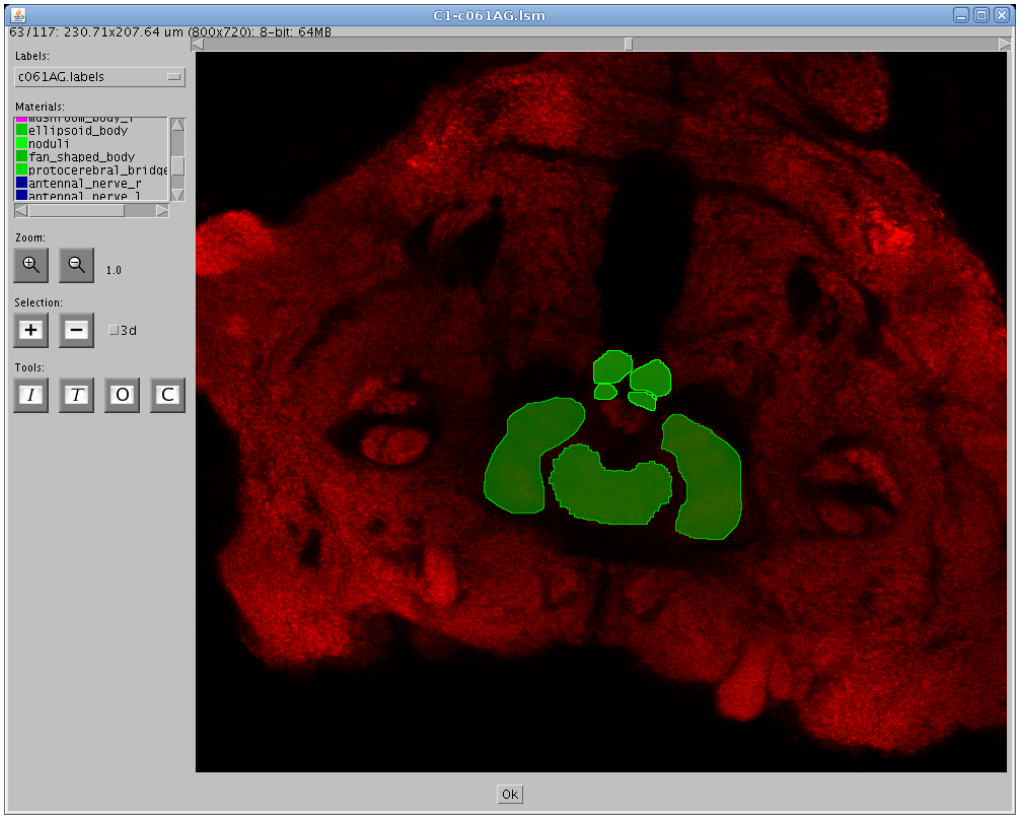


Figure 15 – A screenshot of the `Segmentation_Editor` plugin, showing a single labelled slice from the volume.

of each brain took between 30 and 45 minutes. It is worth noting that this labelling is far from easy in images acquired via confocal microscopy. Many of these neuropil regions have only a thin glial sheath separating them from other regions, so with the inevitable noise in the Z direction they often appear to be "mixed". This is a particular problem for separating the fan-shaped body from the ellipsoid body, the mushroom body from the ellipsoid body and the protocerebral bridge from its surrounding neuropil.

4.4.5 The Computational Morphometry Toolkit

The Computational Morphometry Toolkit (CMTK) is the new name for a set of tools written by Torsten Rohlfing and Calvin Maurer, described in [Rohlfing and C. R. Maurer, 2003]. That paper describes their fast implementation of the algorithm described in [Rueckert et al., 1999], which optimizes a free-form deformation based on cubic B-splines with respect to normalized mutual information. These registrations were run from a script called `munger.pl` which was written by Dr Gregory Jefferis and is distributed from `http://flybrain.stanford.edu`. This method has been successfully applied to the bee brain [Rohlfing et al., 2001] and the *Drosophila* brain [Jefferis et al., 2007], so like the VIB protocol has a published track record of working effectively on images of insect brains.

This algorithm works in two stages: first a global affine transformation is estimated, and secondly an elastic deformation is calculated to refine the global transformation. I found that the affine registration done in the first step often produced poor results, causing the later warping to fail on a large proportion of the images in the corpus. Fortunately, it is possible to provide one's own affine registration based on landmarks, using GNU R code written by Dr Gregory Jefferis and distributed in the `flybrain.stanford.edu` software archive. The results when using this initial affine registration (identical with the **affineo** method mentioned above) are consistently better on this data set so this replacement first step was used in every case.

Although at the start of this work this software was only available in a binary-only form, I heard from Dr Torsten Rohlfing that the intention was that it should be released under an open source license, subject to certain parts being rewritten to not depend on proprietary algorithms. Fortunately, the National Institutes of Health (NIH) funded this work and a version was licensed under the GNU GPL in June 2009. Unlike the other methods studied here, this software is not Java based, but with

the help of Dr Rohlfing I wrote some code for ImageJ that can load the generated transformations, invert them and apply them to points, so the evaluation took place in the same framework as the other algorithms. In the future I hope to work on a front-end for this registration software in Fiji.

4.5 Selection of Landmark Points

A number of the methods we have chosen to test on the image corpus rely on landmark points. This was a conscious bias, because at an early stage in the course of this work, Prof Richard Baldock, an expert in registration of mouse brain images, pointed out to me that the distinctive features that were obvious to the eye in scans of fly brains would be an invaluable starting point for many registration techniques. Following this advice, one of the first steps I took after acquiring the images was to annotate each with named landmark points. Unfortunately, even with the distinctive morphology of neuropil regions in the fly brain, it is not necessarily easy to pick points that are repeatably findable in 3D - for example, many easily identifiable neuropil regions (e.g. the antennal lobes) do not have obvious cusps. After some trial and error, I settled on the following nine points from the mushroom bodies and central complex, which span the volume of the central complex and are relatively easy to find:

- the centre of the ellipsoid body
- the top of the left / right alpha lobe of the mushroom body
- the most lateral part of the mushroom body spur on the left / right
- the most superior point of the left / right part of the protocerebral bridge
- the left / right tip of the protocerebral bridge

These points are shown in Figure 16.

This manual picking of points is, of course, subject to many errors, both from the difficulty of finding the point in noisy scans and the biases of particular annotators. For example, the top of the alpha lobes of the mushroom body are quite large structures, and even trying to maintain consistency with oneself when marking a particular point on this structure is tricky. For the moment, we will simply note that, while this is certainly a problem, even with quite substantial errors these landmarks are useful both as starting points for initial transformations and even as the sole basis of registrations.

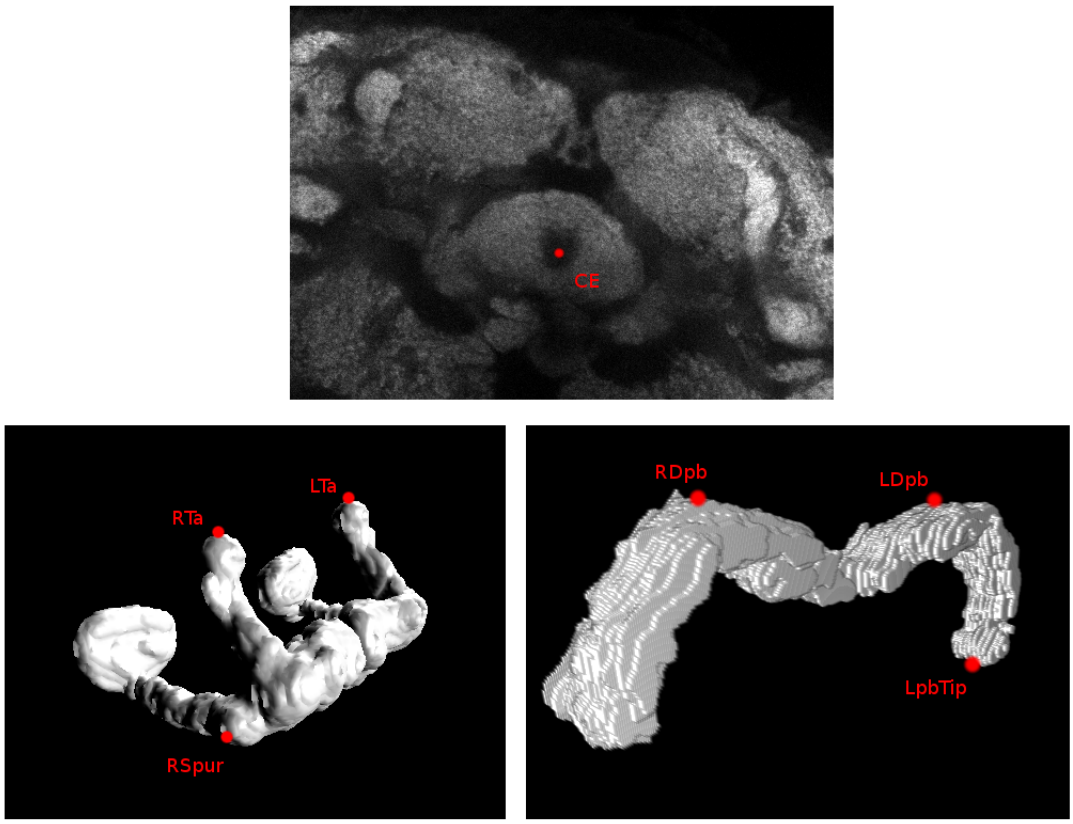


Figure 16 – Chosen landmark points: **CE** = "center of the ellipsoid body", **RTa / LTa** = "top of the right / left alpha lobe of the mushroom", **RSpur** = "the most lateral point of the mushroom body on the right", **RDpb / LDpb** = "the most superior point of the protocerebral bridge on the right / left", **LpbTip** = "the tip of the protocerebral bridge on the left". The top image is a single slice through the ellipsoid body, while the two images below are surface renderings of the neuropil regions extracted from a complete label field, so the points may be more easily visualized.

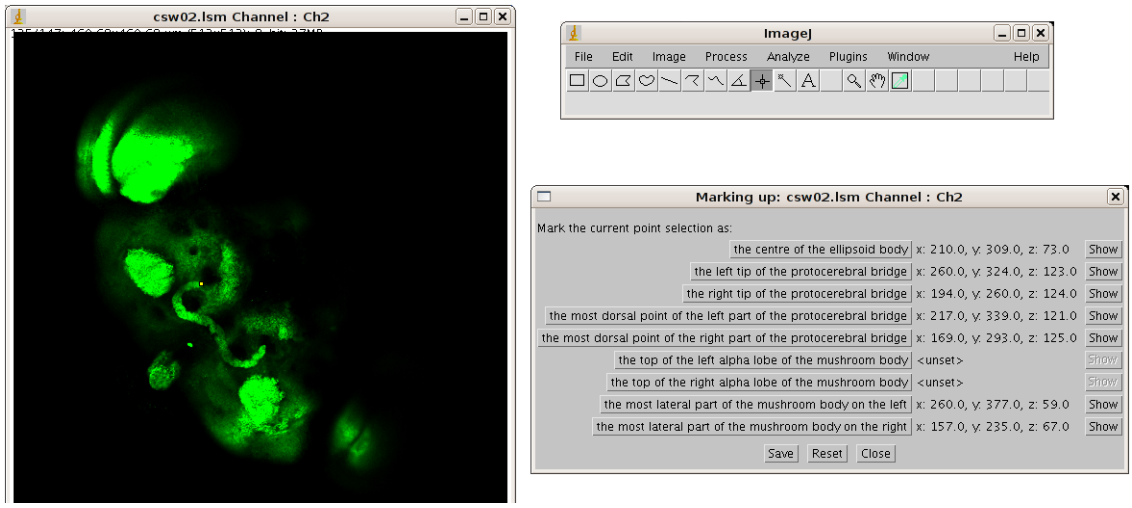


Figure 17 – An early version of the Name_Points plugin. (Later versions offered to fine-tune the point selection based on a template's landmark.)

We will discuss this problem in more detail in sections on refining landmark point selections (section 4.8.2) and automatic landmark point extraction (section 4.8.4).

4.5.1 Name_Points ImageJ PlugIn

ImageJ's support for managing regions of interest is rather lacking for this application, so I wrote a plugin called Name_Points to aid in marking out these points. This plugin is shown in Figure 17. In several of the images in the corpus it was impossible to pick out one or more of these points, so not all images have the complete set of points.

I have developed a more generally useful version of this plugin, retaining the original functionality, which is bundled in Fiji as the plugin "Name Landmarks and Register". I am grateful for contributions from Dr Gregory Jefferis to the development of this plugin, which are detailed in the repository history.

4.6 Choosing a Template Image

Each time we register a new image, it must be done so against a template of some kind. The template should be chosen to be an "ideal" image in the sense that it has as few distortions as possible. The template used for image registration applications is sometimes a mean of many registered images,

but even in that case the source images which are used in the average must have been registered against one distinguished image before the averaging. Some of the properties one looks for in a template image are:

1. The scan should be of excellent quality, with sharp edges to the different neuropil compartments.
2. Ideally, the template should not be cropped too closely around the neuropil regions of interest, since some registration algorithms (e.g. CMTK) perform notably less well at the edges of the image.
3. The scan should have as few distortions as possible. This is somewhat difficult to quantify, but we choose to do so by measuring the quality of fit given by rigid + scaling transformations generated solely from landmarks in the template image and the model images. This is explained in more detail below.

As was discussed earlier in this section I have manually annotated each image with landmark points. Any two sets of landmark points can be used to define a mapping; for evaluating the images to find a candidate template brain, we find the best rigid transformation between each image and the candidate template. The images which have the mean best fit against all the other images in the corpus will be the most typical of our data set. We also compare each image to an independent brain, the Canton41c image used by Dr Arnim Jenett. In each of these sets of comparisons we evaluate the registrations with three different measures: mutual information between the images, correlation between the images and finally the mean distance moved by the landmark points under the transformation.

We find that one image, c005BA, appears in the top 12 brains under any of these evaluation criteria, as can be seen in Figures 18 to 23, in which the name of that image is shown in bold. In each graph, the best candidates are towards the right hand side of the x-axis (i.e. bottom of the page in portrait orientation). I confirmed by eye that the c005BA brain is indeed one of the best nc82 images in this corpus, with no obvious distortions. The image did have some extraneous material above the brain, so I manually edited this out before using the image for future registration. The images in the corpus (mostly acquired before our group added a rotating stage to its confocal microscope) have the major axis of the brain lying at random angles within the image volume - the major axis of the c005BA brain is about 15° from the y axis.

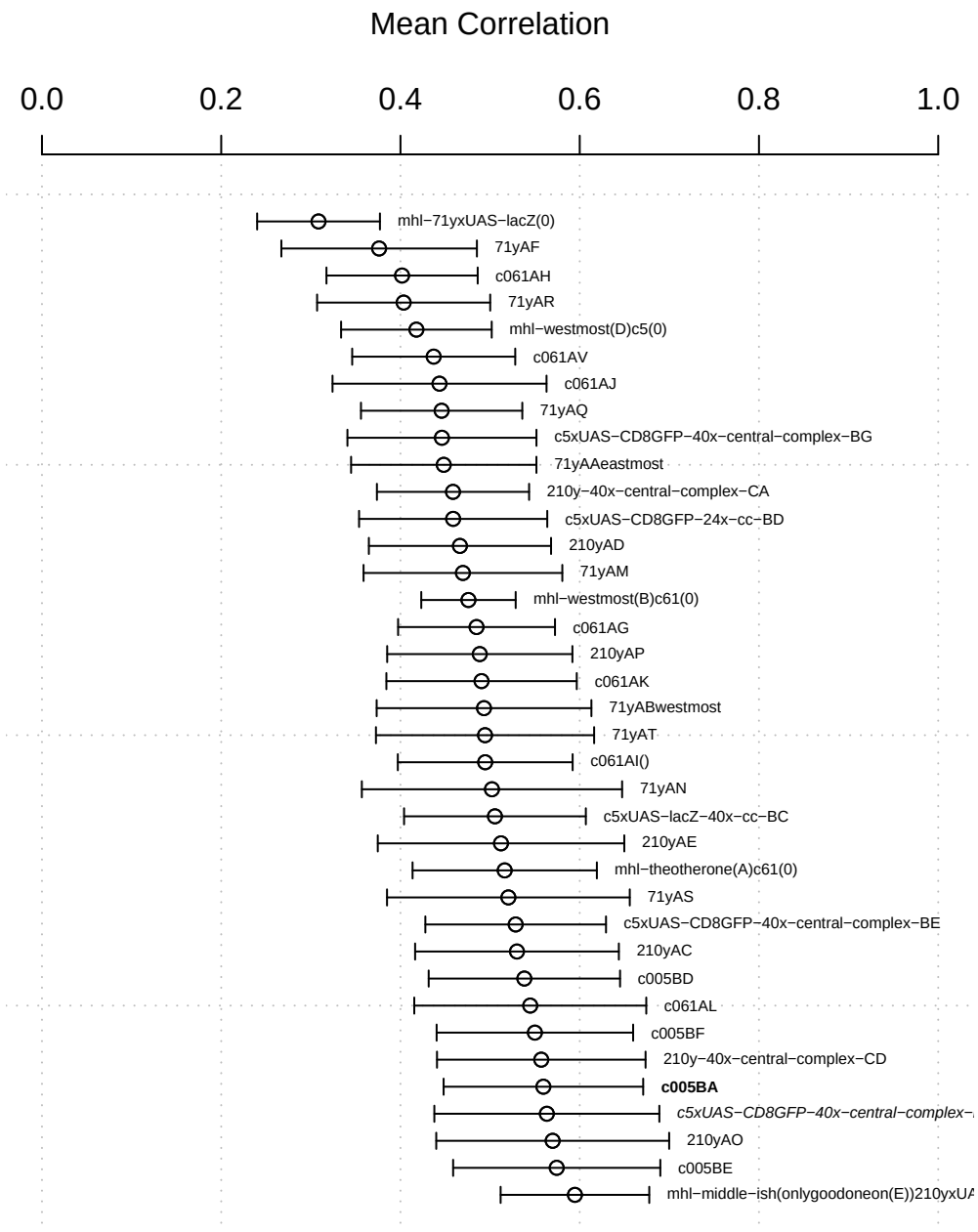


Figure 18 – The mean correlation between each corpus image and the rest of them under the best rigid transformation between their landmark points. (In this, as in the subsequent graphs, the image picked out in bold (c005BA) is the best template overall, while that in italics is the next best candidate.)

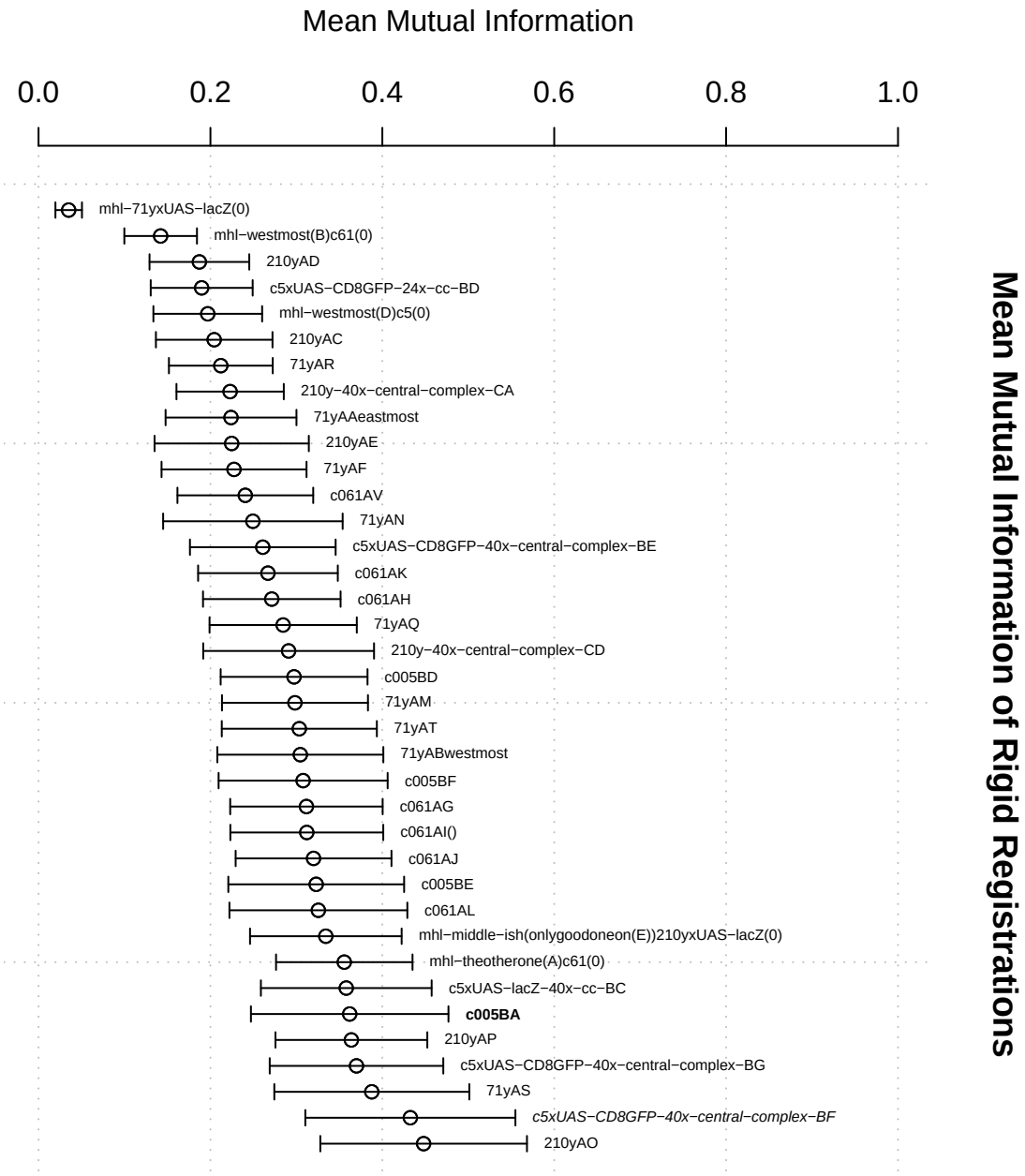


Figure 19 – The mean mutual information between each corpus image and the rest of them under the best rigid transformation between their landmark points.

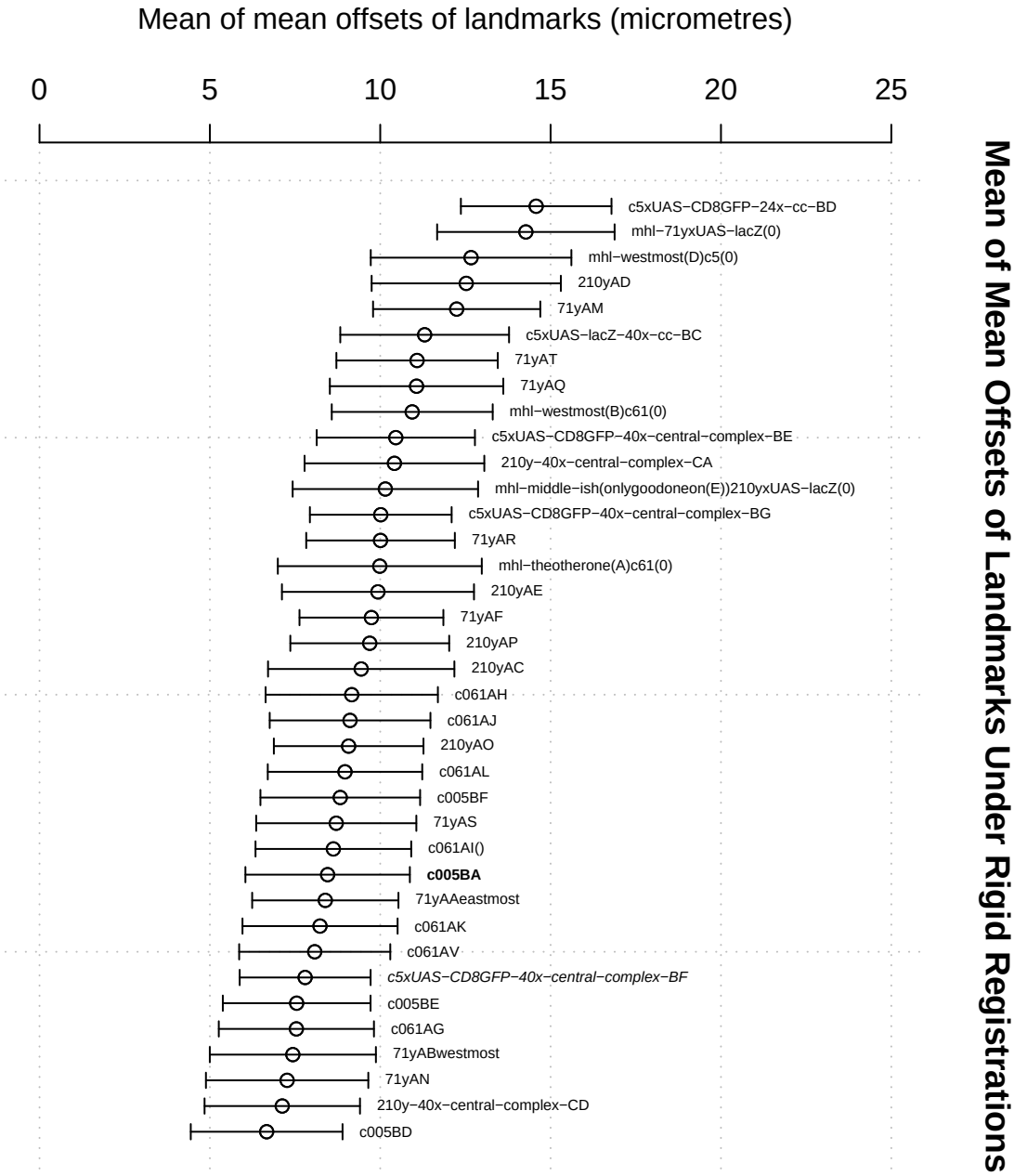


Figure 20 – The mean of the mean distances moved by the landmark points under their best rigid mapping between each corpus image and the rest of them.

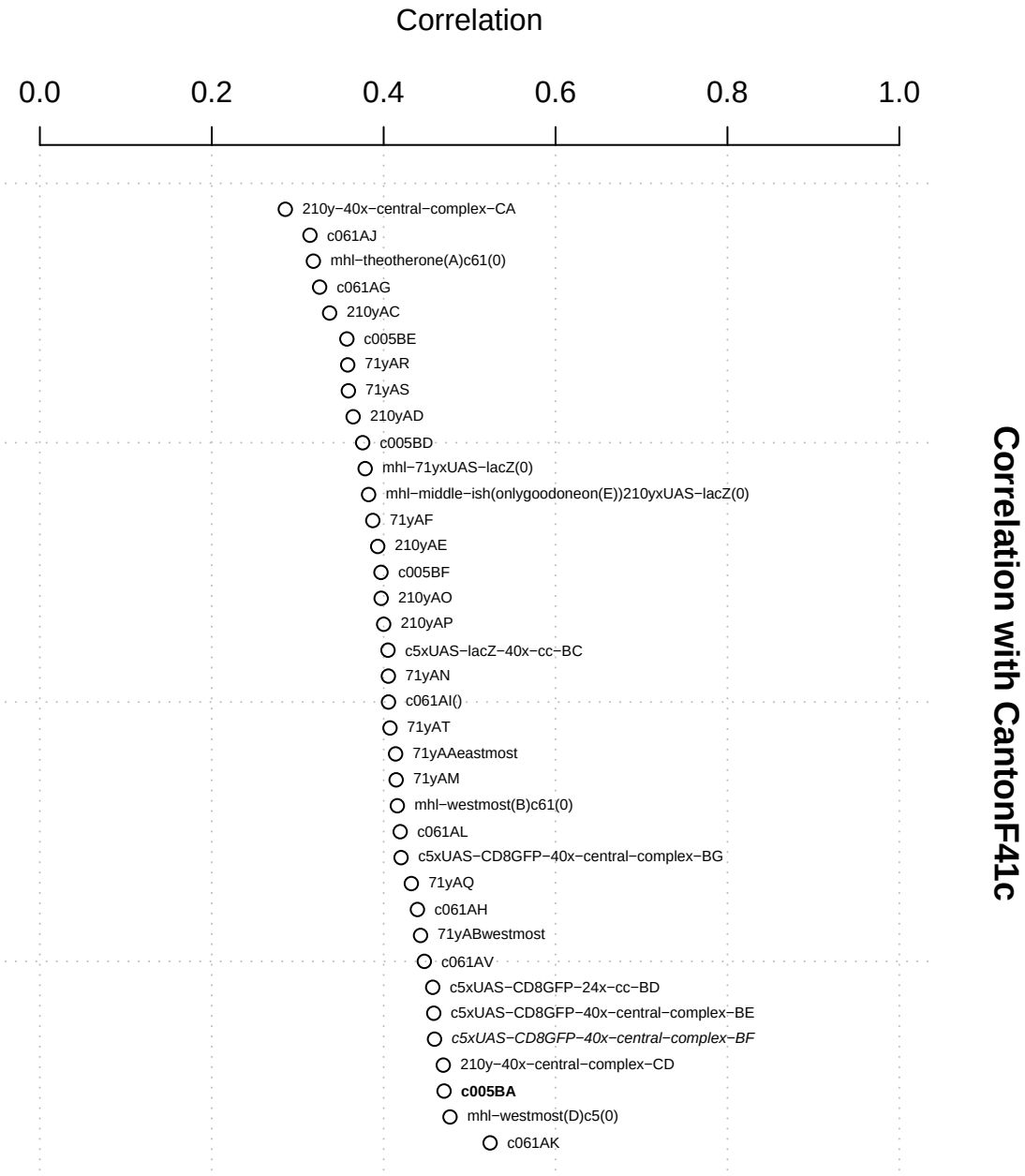


Figure 21 – The correlation between each image and CantonF41c under the best rigid mapping between them.

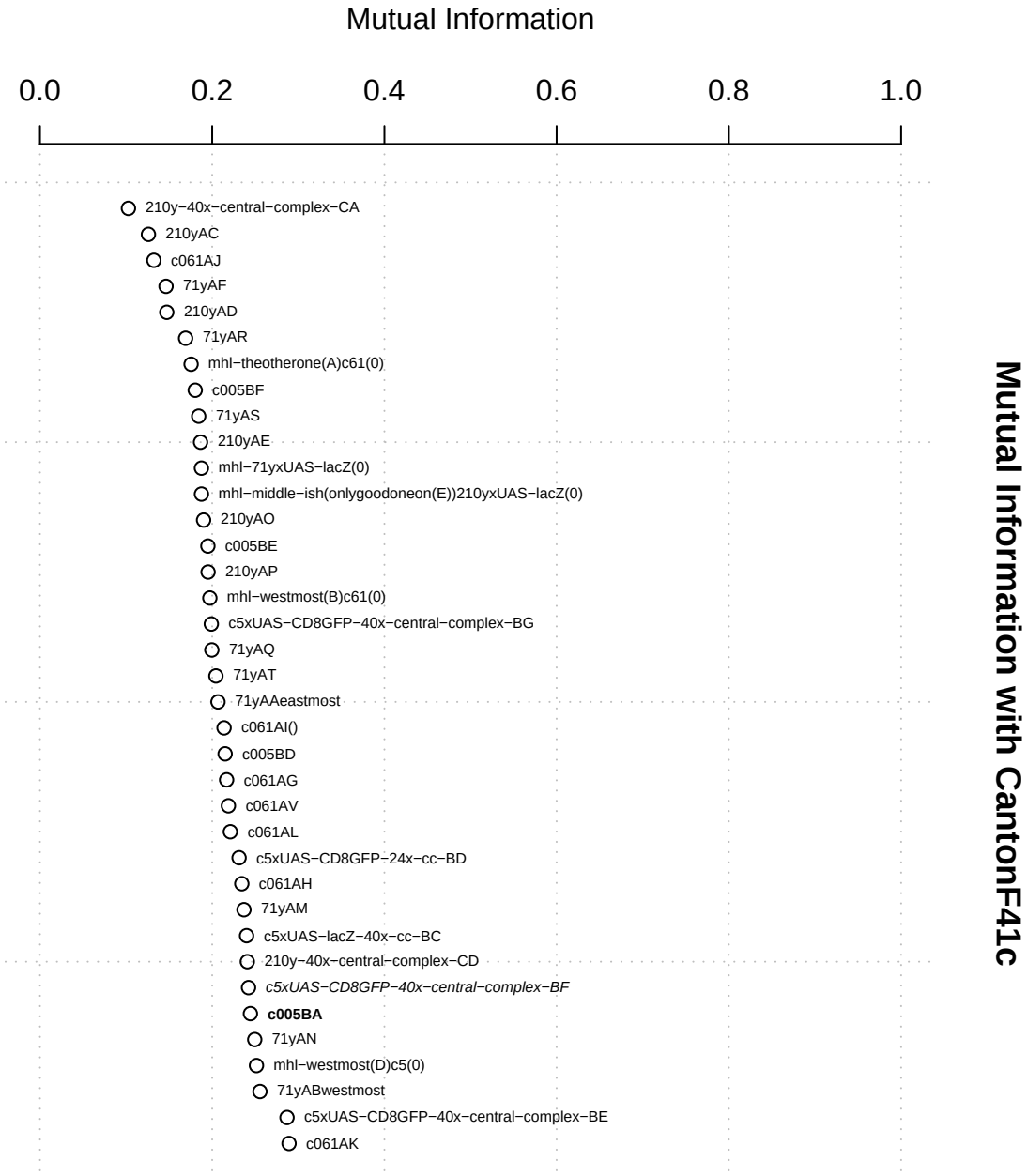


Figure 22 – The mutual information between each image and CantonF41c under the best rigid mapping between them.

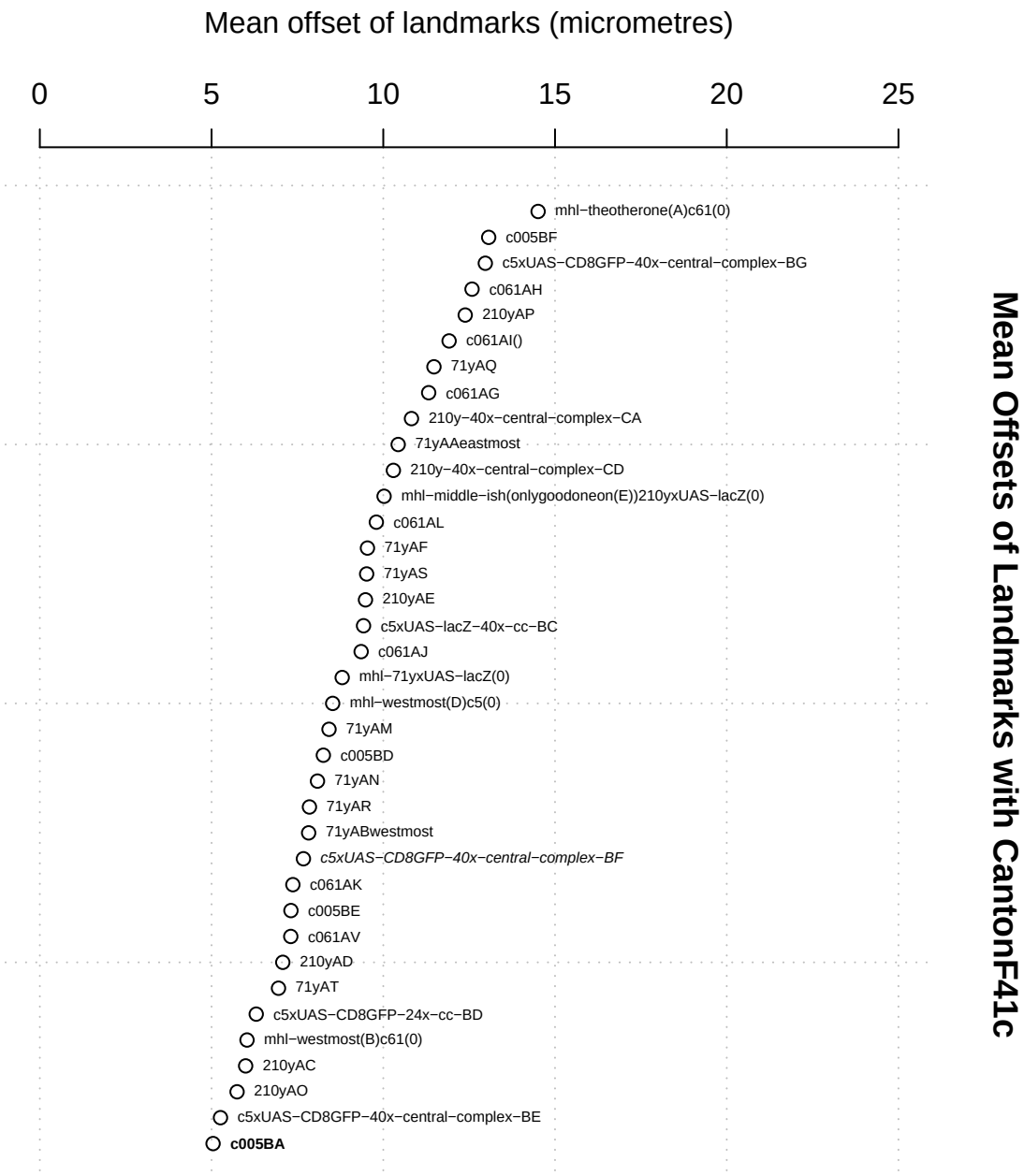


Figure 23 – The mean distances moved by the landmark points under their best rigid mapping between each corpus image and CantonF41c.

In summary, I chose to use this brain, c005BA, as the template in most of the work that follows. However, in the next section we consider scoring registrations against both c005BA and an averaged template, to see whether this affects the results. The averaged template used in this comparison was generated from the results of registering every brain against c005BA with the CMTK method and averaging the registered brains, as described in section 4.9. (The CMTK method turns out to produce the best registrations by any of the measures we used, as will be shown in section 4.8.)

4.7 Evaluating Registration Quality

In evaluating registration algorithms, it is common practice to compare the results on the basis of quantitative measures of the quality of the alignment of the images, such as mutual information or correlation. While it is certainly the case that perfect alignments will score very well on these measures, it was unclear to us whether these measures are good proxies for the evaluation that we are really interested in. This ideal criterion might be characterized as:

- Which registrations are more useful to a neuroanatomist trying to identify corresponding positions within the central complex?

The most important points to note about this are that (a) it is defined in terms of a human expert's idea of usefulness and (b) it is limited to a particular region of interest.

It is impractical (and rather unreliable) to rely on human experts to evaluate very large numbers of registrations. Instead, we chose to try to establish how well the judgements of neuronanatomists agreed with the decisions made by a variety of automatic measures, and hoped to establish that one or more of these measures is reasonable as a proxy for a human classifier.

4.7.1 Definitions of Measures

I consider three typical measures of image similarity in this section, each of which is defined on pairs of values. If we assume that the set of the position vectors of all points in the image is Ω (of size n), then the sample intensity in the model is given by the function $M : \Omega \rightarrow [0, 256)$ and the sample intensity in the template is given by $T : \Omega \rightarrow [0, 256)$, then the measures are defined as follows:

Euclidean: The "Euclidean" measure is defined as:

$$\sqrt{\sum_{x \in \Omega} M(x)^2 + T(x)^2}$$

... so called because if we consider all the values in each image to be vectors of dimension n then this measure is the standard Euclidean distance between those vectors. We abbreviate this measure below as "E" when the template image is c005BA and "E-S" (for Euclidean Smoothed) when the template image is the averaged image.

A possible objection to this simple measure is that it is sensitive to the distribution of values in each image. For example, even if a perfect registration was found between two images, the registration would be scored differently depending on the PMT settings used to acquire them in the first place. It is possible to normalize the images beforehand to compensate for this, but I have chosen not to do so since it is possible that the human experts' judgements are similarly affected by differences in contrast and brightness.

Correlation: The correlation measure is defined as:

$$\frac{\frac{\sum_{x\in\Omega} M(x)T(x)}{n} - \frac{\sum_{x\in\Omega} M(x)\sum_{x\in\Omega} T(x)}{n^2}}{\sqrt{\frac{\sum_{x\in\Omega} M(x)^2}{n} - \frac{\left(\sum_{x\in\Omega} M(x)\right)^2}{n^2}}\sqrt{\frac{\sum_{x\in\Omega} T(x)^2}{n} - \frac{\left(\sum_{x\in\Omega} T(x)\right)^2}{n^2}}}$$

Strictly speaking this is the Pearson product-moment correlation coefficient; if we consider M and T as random variables, and say that $E(X)$ and $var(X) = E(X^2) - (E(X))^2$ are the expectation and variation of the random variable X, then it could be written more clearly as:

$$\frac{E(MT) - E(M)E(T)}{\sqrt{var(M)}\sqrt{var(T)}}$$

Mutual Information: We define the joint probability $p(a, b)$ to be the number of points x such that $M(x) = a$ and $T(x) = b$, divided by n. The marginal probability $p_M(a)$ is the number of points x such that $M(x) = a$ divided by n, and similarly $p_T(a)$ is the number of points x such that $T(x) = a$ divided by n. Then the mutual information is:

$$\sum_{a\in[0,256)} \sum_{b\in[0,256)} p(a,b)\, log_2\left(\frac{p(a,b)}{p_M(a)p_T(b)}\right)$$

4.7.2 Templates

Although the human evaluators were only asked to evaluate registrations based on overlays of model images on the c005BA image, we also consider automatic evaluations based on registrations against the averaged template. I decided to include these additional methods since the averaged template is naturally much less noisy than the c005BA image, and it is possible that this would cause the automatic measures to perform less well as registration evaluators.

As can be seen from Figures 24, 25 and 26 the relationships between each measure against c005BA and the averaged image are approximately linear, but clearly not monotonic, so continuing to consider the evaluation against both templates may be worthwhile.

4.7.3 Registration Region of Interest

It is possible to simply apply our automatic measures to every point in the template and every corresponding point in the transformed image, but since the brains may be at random orientations we will have to somehow deal with points that have no correspondence - that is those template points that under a given mapping would be mapped back to somewhere outside the transformed image. If we ignore these, a naïve measure may give good scores to registrations that cause only a small region of the images to overlap. On the other hand, if we include these points, then we may penalize otherwise good registrations for missing regions due to the angle of the brain at acquisition. Some measures, such as normalized mutual information, take this into account, but the simplest approach is to define a region of interest in the template brain, and only score the registrations based on the mapping of points within this region, regardless of the measure. We select this region such that it is completely present in as many subject brains as possible.

The focus of this investigation is on neuroanatomy of the central complex, so, at a minimum, the region of interest I defined for scoring registrations must be a convex hull of the fan-shaped body, ellipsoid body, noduli and protocerebral bridge. These are all contained within the acquired images, although in the case of the protocerebral bridge, the images are often cropped very close to that structure. (This is because the natural way for a dissected fly brain to lie on a microscope slide is with the occipital plane downwards - thus we tend to have the lowest signal around the protocerebral bridge, which is very close to this surface.) In addition, the images almost all contain the extreme lateral and superior points of the mushroom bodies so I extended the region of interest to be a cuboid region that included these. These points are also used as landmarks, so the landmark-based registration algorithms will be well informed about the position of these structures. Slices through the template image showing only the points in the region of interest can be found in Figure 27.

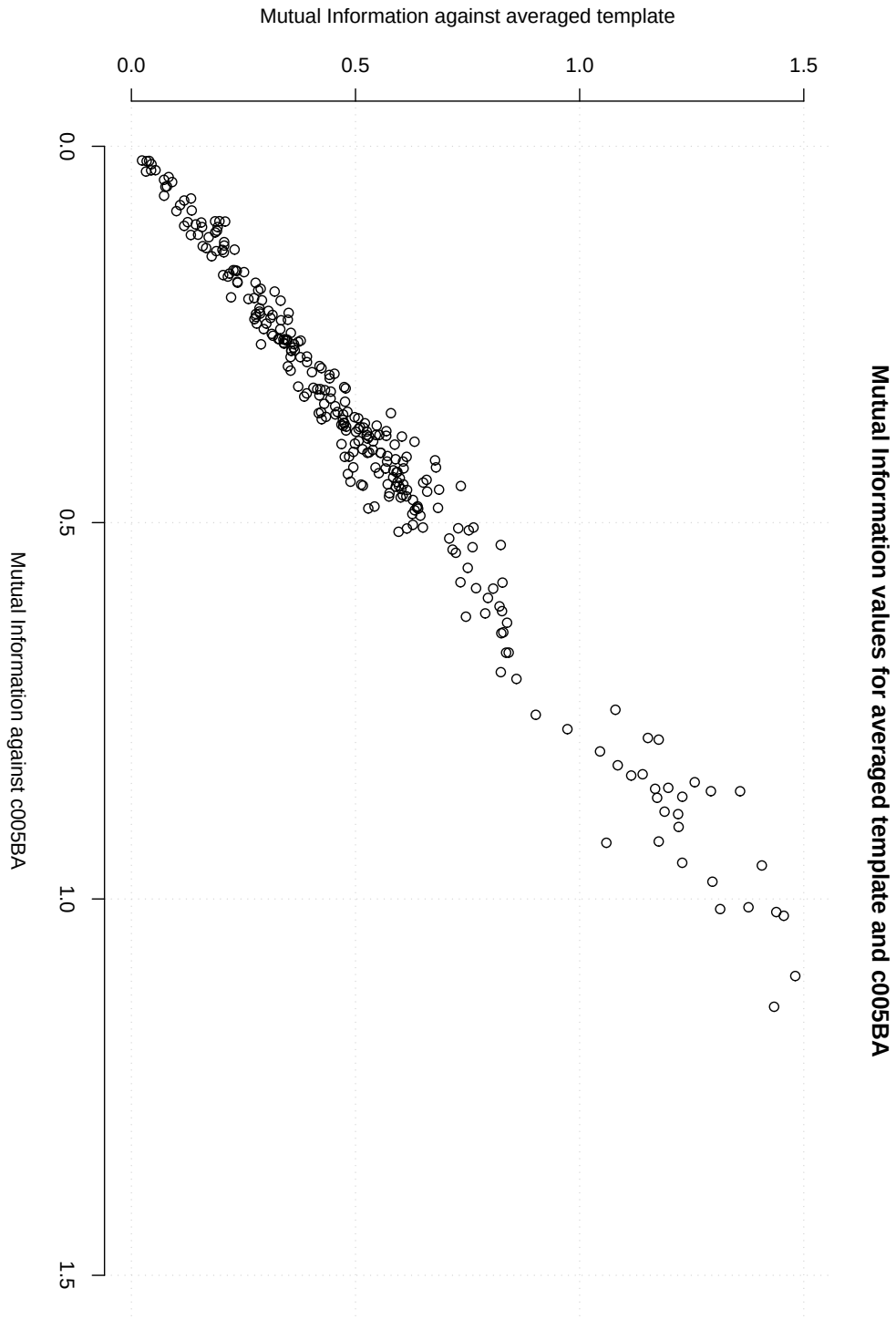


Figure 24 – Comparisons of mutual information between registered images and either the c005BA template or the averaged template.

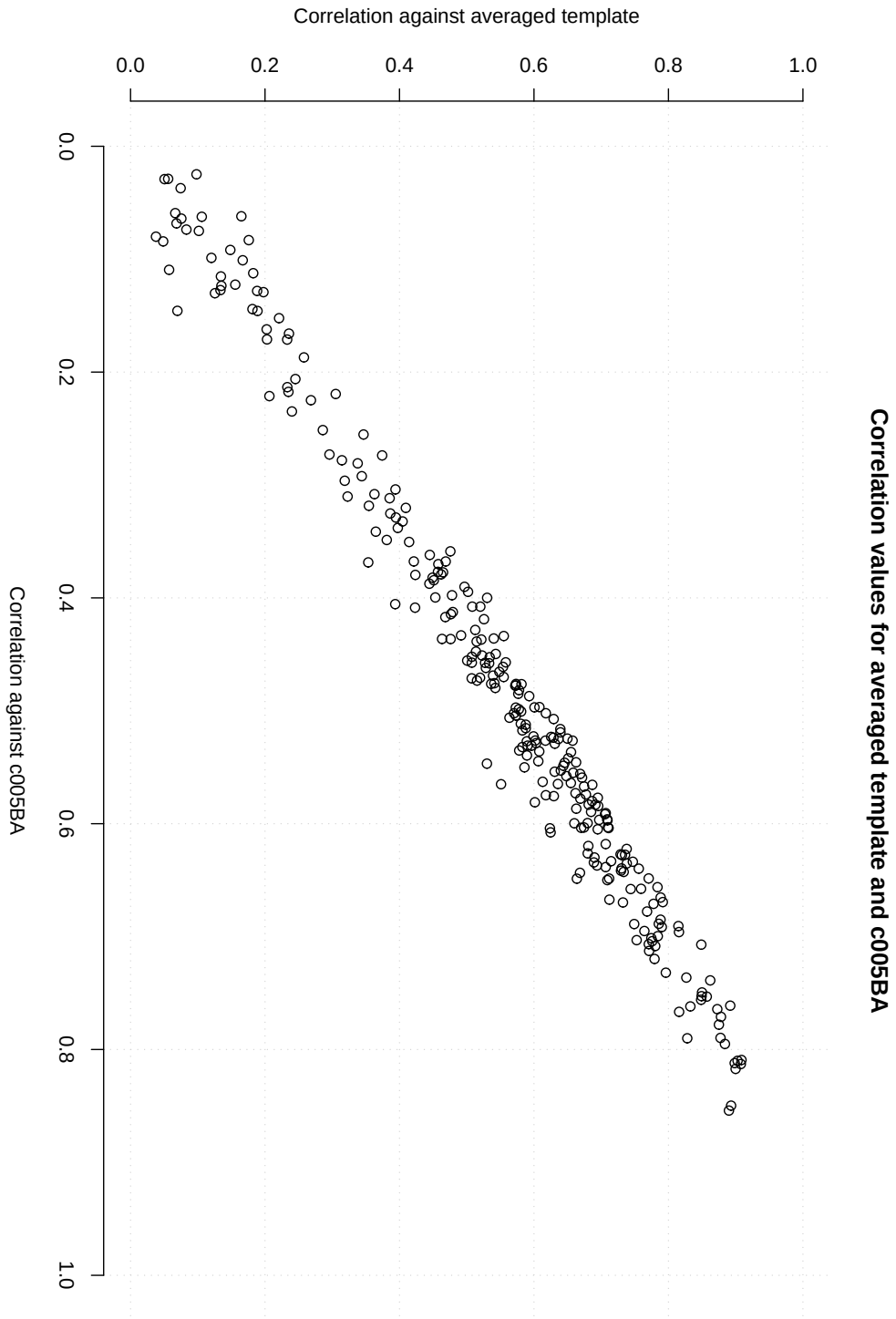


Figure 25 – Comparisons of correlation between registered images and either the c005BA template or the averaged template.

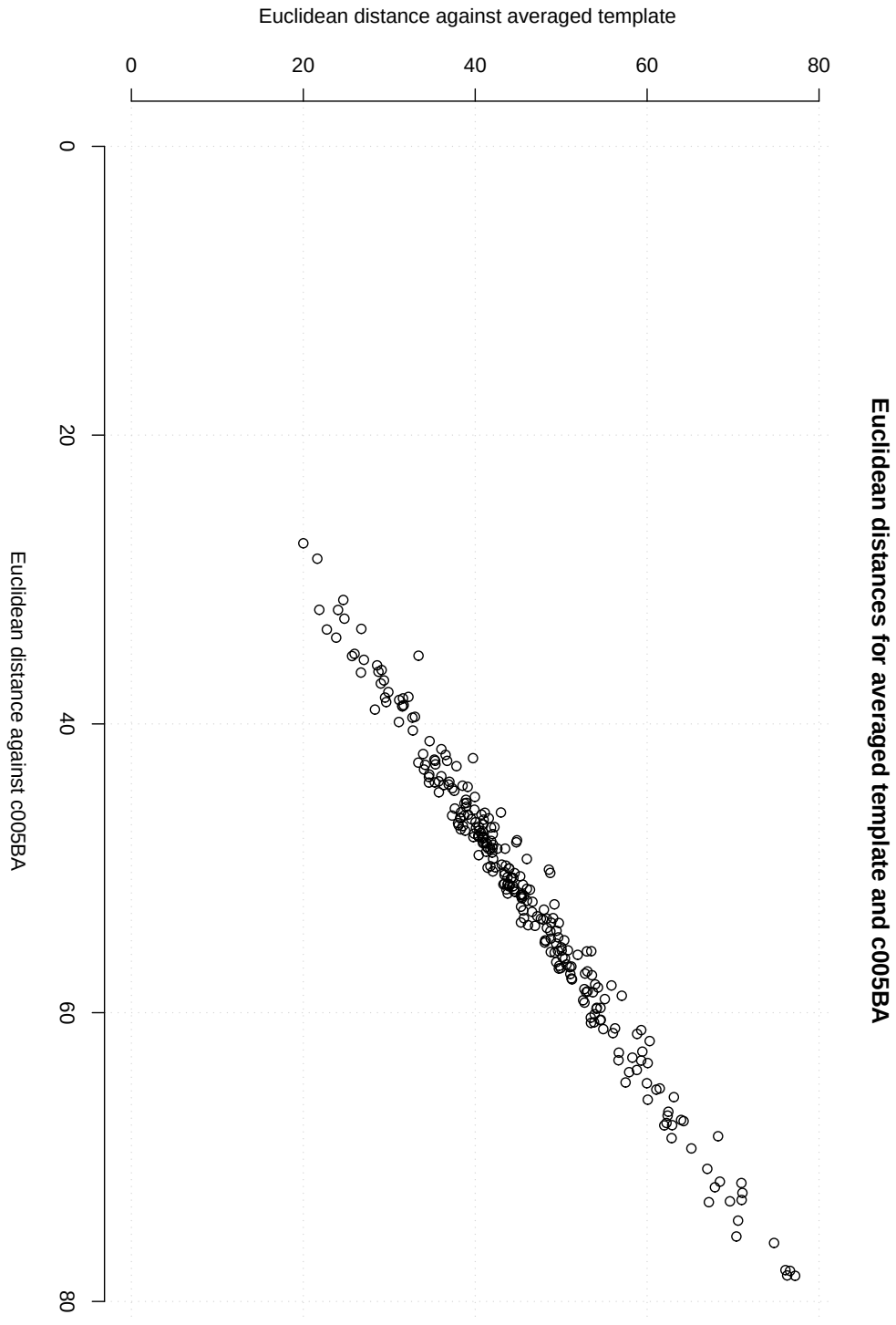


Figure 26 – Comparisons of Euclidean distance between registered images and either the c005BA template or the averaged template.

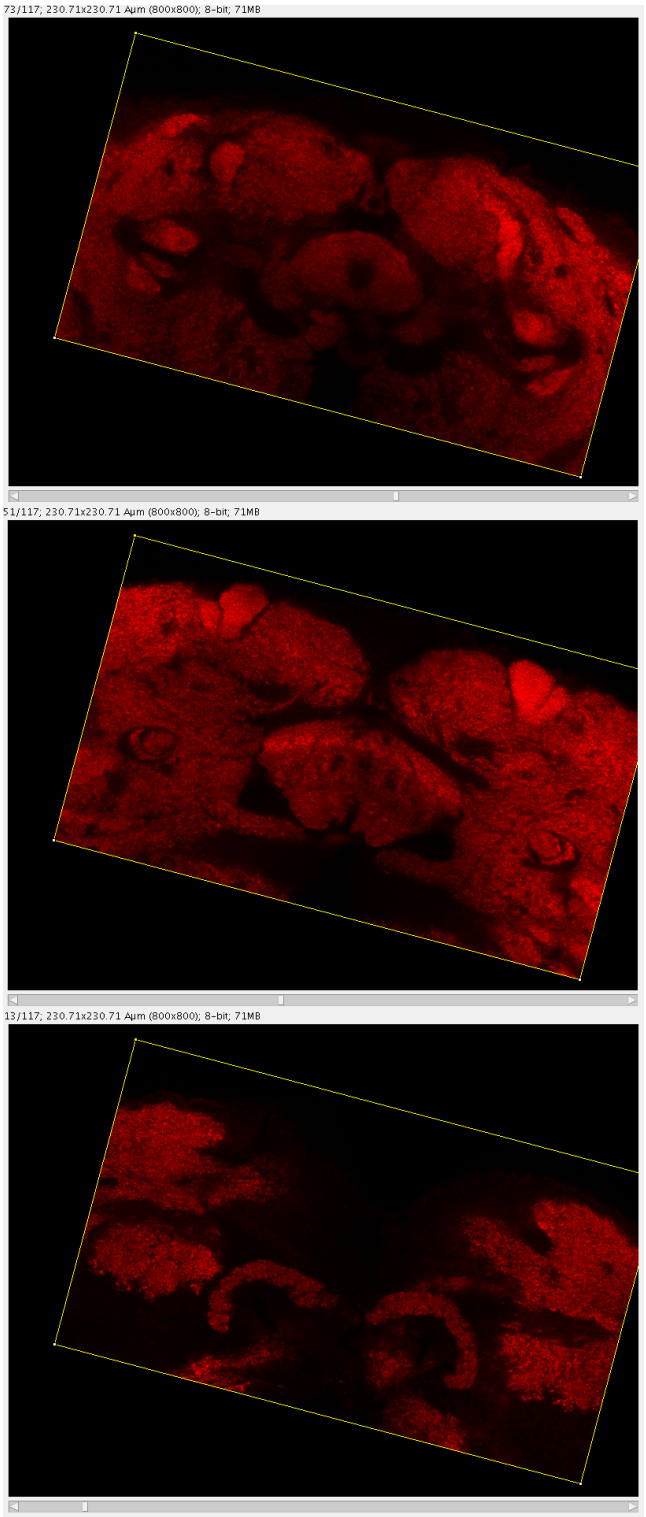


Figure 27 – Slices through the ellipsoid body, fan-shaped body and protocerebral bridge, showing the cuboid region of interest used for scoring registrations.

4.7.4 Pairwise Comparisons by Human Experts

After some experiments (e.g. in Chapter 6) in which I tried to rate the quality of registrations on a scale from 1 to 10, it became apparent that it was very difficult to maintain a consistent scale and thus this type of rating was not an ideal basis for comparison. Instead, I adopted a system of pairwise comparisons, as frequently used in psychology to assess preferences.[37]

In order to present the candidate images and record the experts' preferences, I wrote an ImageJ plugin called "Pairwise Comparisons". This presents two image stacks side-by-side, and waits for the user to select one by clicking on it. Each of the two image stacks represents a registration, and shows a transformed model image overlaid onto the template. A screenshot of this interface is shown in Figure 28.

The four experts who used this system to record their preferences were myself[38] and three other members of the Armstrong group, all of whom were experienced in examining confocal stacks of the central fly brain. Each expert was given the instructions to scroll through the stacks and click on the one that would be most useful for comparing neuroanatomy of the central complex, and in particular the fan-shaped body and ellipsoid body. For cases in which it was difficult to decide based on those regions, the experts were told to next consider the alignment of the protocerebral bridge and the mushroom bodies in order to make a decision.

The images that were presented were selected pseudo-randomly (based on a consistent seed) from all possible pairings of corpus images registered with the **affine**, **tps** and **rigid** methods. This provided a good range of qualities of registrations.

The choices of the four experts ("L", "A", "Y" and "K") are shown as the bottom four horizontal strips in Figure 29. Within the group of experts, agreement between any two of them ranged from 63% (A and Y) in the worst case to 82% (L and K) in the best case, where one would expect 50% agreement from two assessors choosing completely at random. The rather low figure in the worst case reflects the reality that deciding which registration is better between any randomly chosen pair

[37] In psychology, the use of pairwise comparisons was pioneered by L. L. Thurstone. However, perhaps the most widely known application is in "picture battle" websites such as `http://www.kittenwar.com` which attempts to order user-submitted pictures of kittens according to "cuteness" by repeatedly presenting visitors with randomly chosen pairs of images and asking them pick the cuter kitten.

[38] Although it is far from ideal to have the experimenter included in the experts expressing preferences, the images were selected at random and displayed no evidence (other than perhaps occasional characteristic distortions) of the method used to generate the registration.

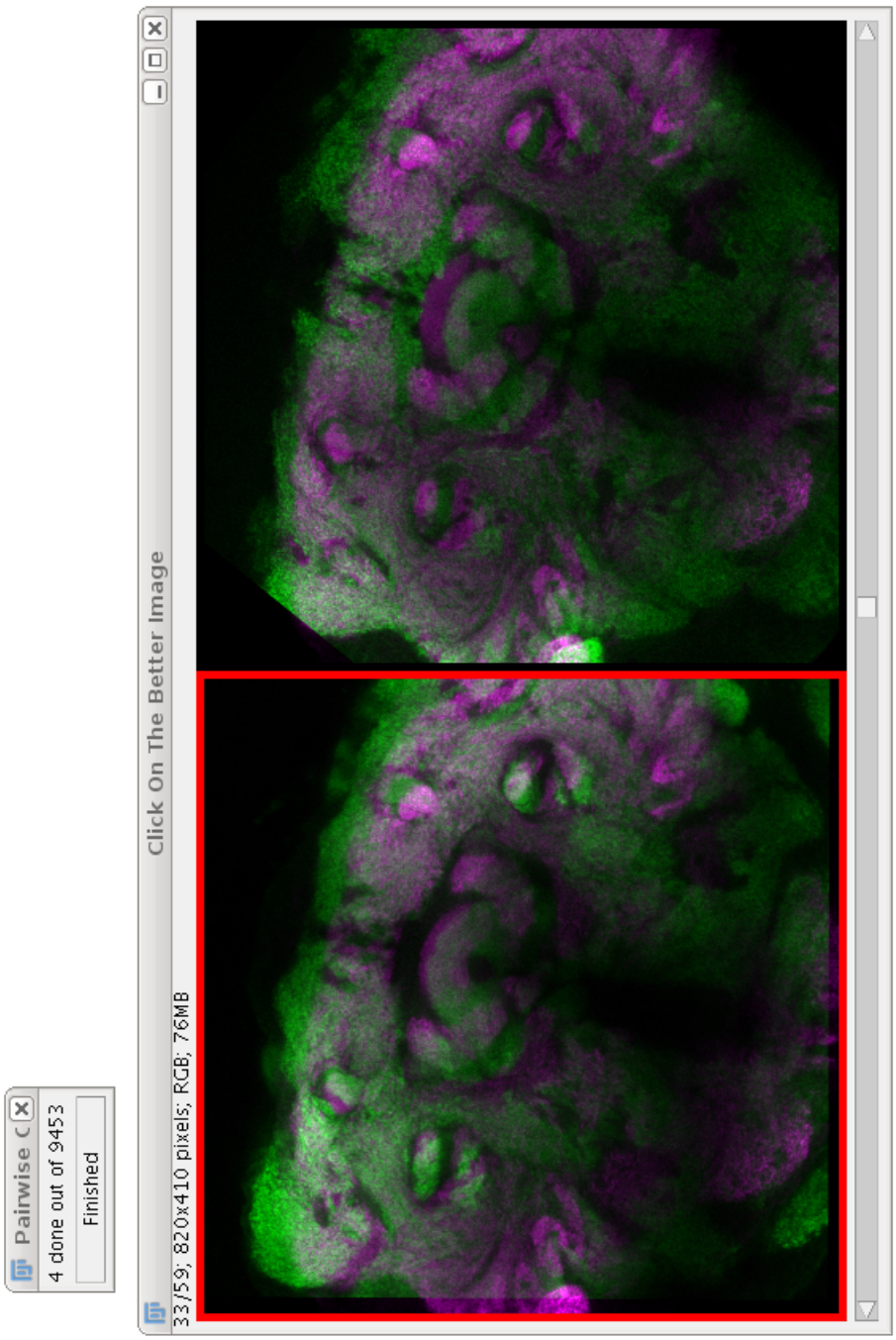


Figure 28 – A screenshot of the Pairwise Comparisons plugin presenting two candidate registrations. The red border around the image on the left is a highlight that shows which image the mouse is over, and hence which will be selected on clicking.

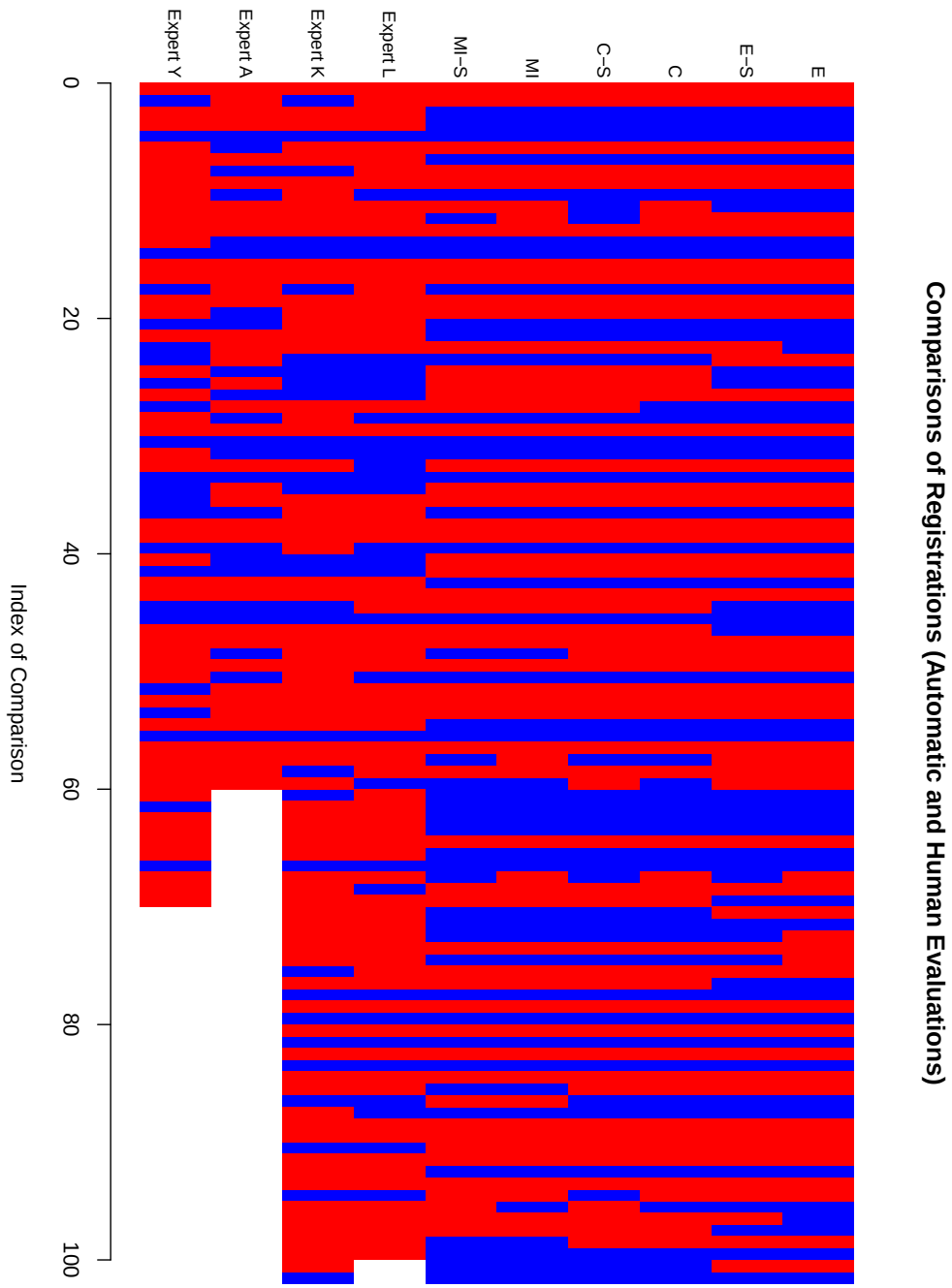


Figure 29 – Pairwise comparisons of registered and overlaid images. The experts "L", "K", "A" and "Y" were each asked to pick the better registration of two randomly but predictably chosen images. If they picked the right hand image, this is represented by a red block; if they picked the left hand image, this is represented by a blue block. The top six strips show the corresponding decisions made by a variety of quantitative measures: "E" for "Euclidean", "C" for "Correlation" and "MI" for "Mutual Information". The suffix "-S" means that the score against the smoother averaged template was used, rather than the more noisy c005BA image.

Method A	Method B	Agreement
MI	MI-S	98%
E	E-S	97%
C	MI	96%
C-S	MI-S	95%
C	C-S	95%
C	MI-S	94%
C-S	MI	91%
C	E-S	88%
C-S	E-S	87%
C	E	85%
E-S	MI	84%
C-S	E	82%
E-S	MI-S	82%
E	MI	81%
E	MI-S	77%

Table 3 – Degree of agreement between the automatic evaluators. (As we would expect, the highest agreements are between the same measure with different templates.)

is genuinely difficult. In particular, this test set included many pairs where both registrations were quite poor, in which case it is not clear that an informative comparison can be made.

As expected (and shown in Table 3) the automatic measures agree with each other on which registration is better in a much higher proportion of cases.

Figure 30 shows the percentage agreement between each expert and each automatic measure. The most obvious feature of this graph (from the vertical bands) is that some experts have a higher level of agreement with the automatic measures than others. Another clear point is that there is no definitively "best" automatic measure that matches all the human experts. Each expert seems to have different preferences - for example:

- Expert Y uniquely agrees more strongly with measures against c005BA than against a smoothed template (indicating perhaps that this expert put more emphasis on the comparative grey

Figure 30 – A 2D histogram showing the percentage agreement between every expert and every automatic measure.

values, whereas the others looked more at the overlap).

- Expert A's assessments match with the MI measure best, whereas the others prefer one of the correlation or Euclidean metrics.

However, it should be noted that the 95% confidence intervals (calculated using the Wilson score interval) for the values in this table are between ±9% and ±12%, so these differences are unlikely to be statistically significant.

It should also be noted that these levels of agreement are all quite low - only just above chance in some cases. An obvious problem with considering these data purely in terms of proportion of cases in which two evaluators agree is that it doesn't allow us to see the extent to which the ability to compare registrations is compromised by either (a) the registrations being of similar (and possibly very good) quality or (b) neither registration being good enough to use.

We can visualize these issues by plotting 2D histograms where the colour of a box represents the level of agreement between the automatic measure and all human evaluations of the pairs that fall into that bin. However, since the distribution of these scores is far from uniform, it is important to take note of the confidences intervals and values of n for each box in the figures that follow. The value of n and the 95% confidence intervals (calculated with Wilson's method) are given in the text in each box. In addition, the side of each box is linearly scaled with the width of the confidence interval to give a visual indication of how much confidence can be had in that result, smaller boxes representing less certain values.

Firstly, in Figures 31 to 36, we bin the pairs of scores on two axes by:

- *(y-axis, corresponding to consideration (a))* The absolute difference in scores. When this is low, suggesting the registrations are of very similar quality, we would expect poor levels of agreement (around 50%).

- *(x-axis, corresponding to consideration (b))* The best score of the pair. When this value is high, that should logically exclude all pairs where both of the registrations are evaluated as being very poor, so we might expect there to be more agreements.

The results shown in these plots are somewhat surprising. The measures that seem to correspond best with our expectations about how a good automatic measure should correspond to human judgements

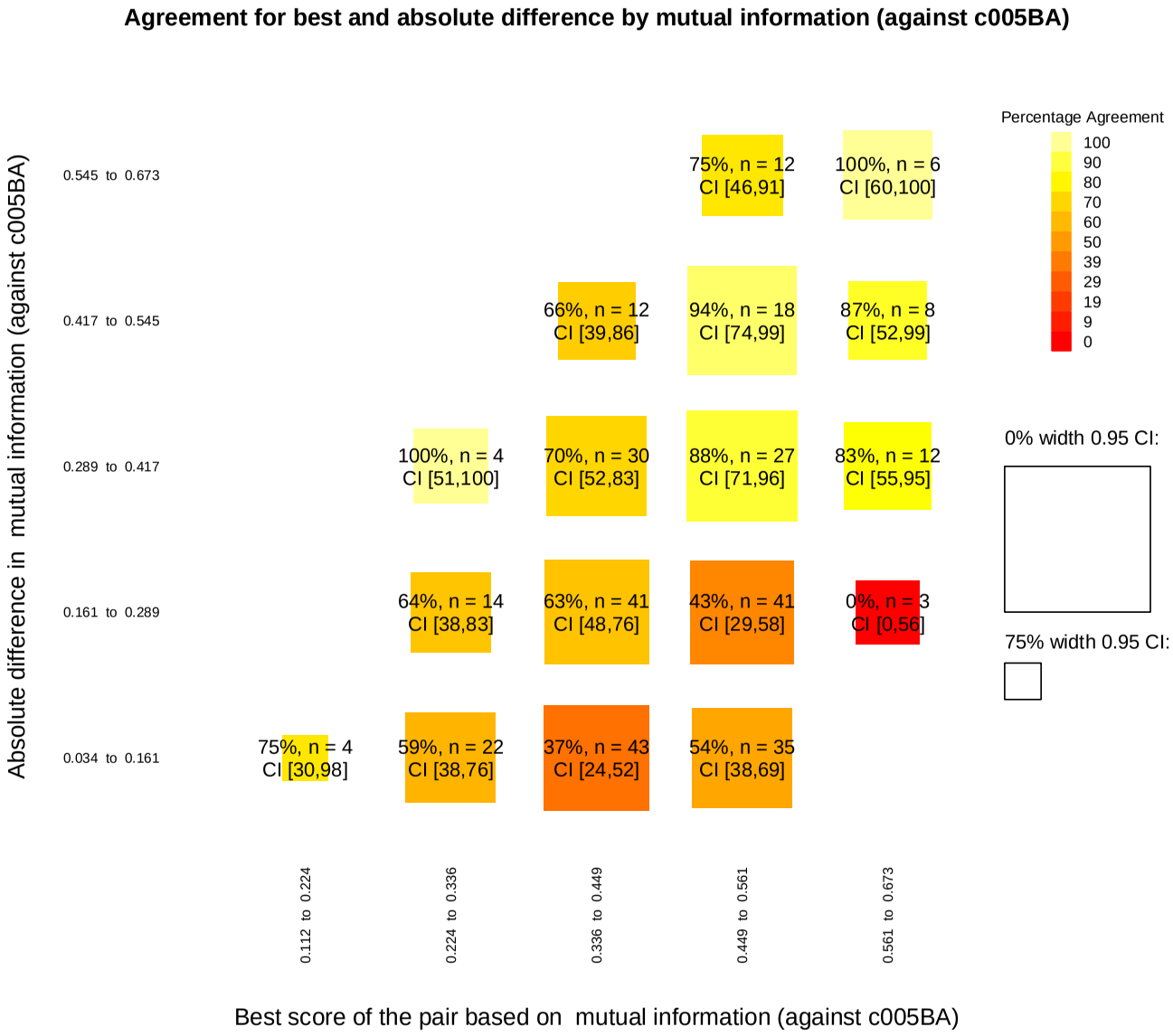


Figure 31 – Agreement with human evaluators of the MI measure, binned by best score on the x-axis and absolute difference in scores on the y-axis.

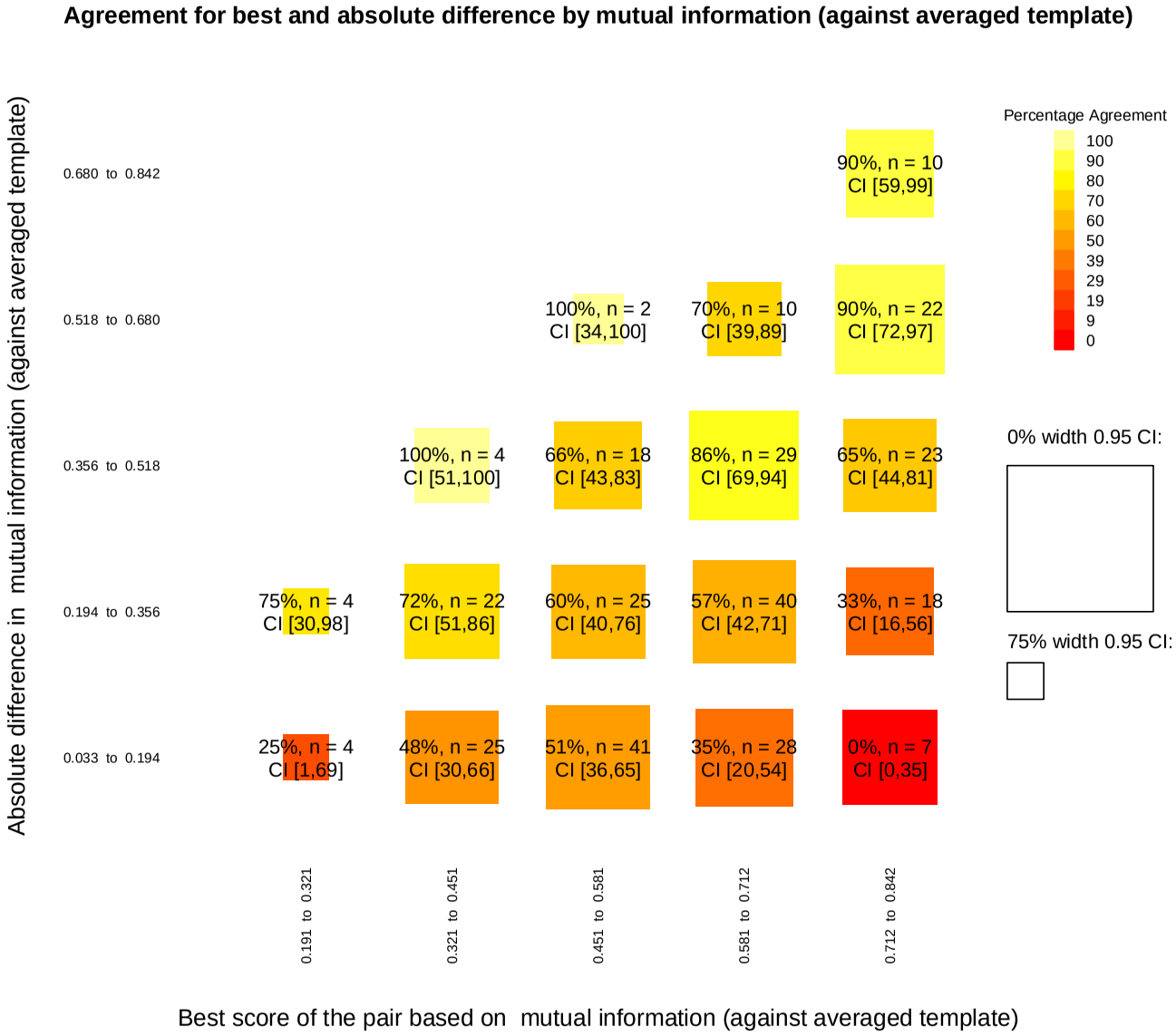


Figure 32 – Agreement with human evaluators of the MI-S measure, binned by best score on the x-axis and absolute difference in scores on the y-axis.

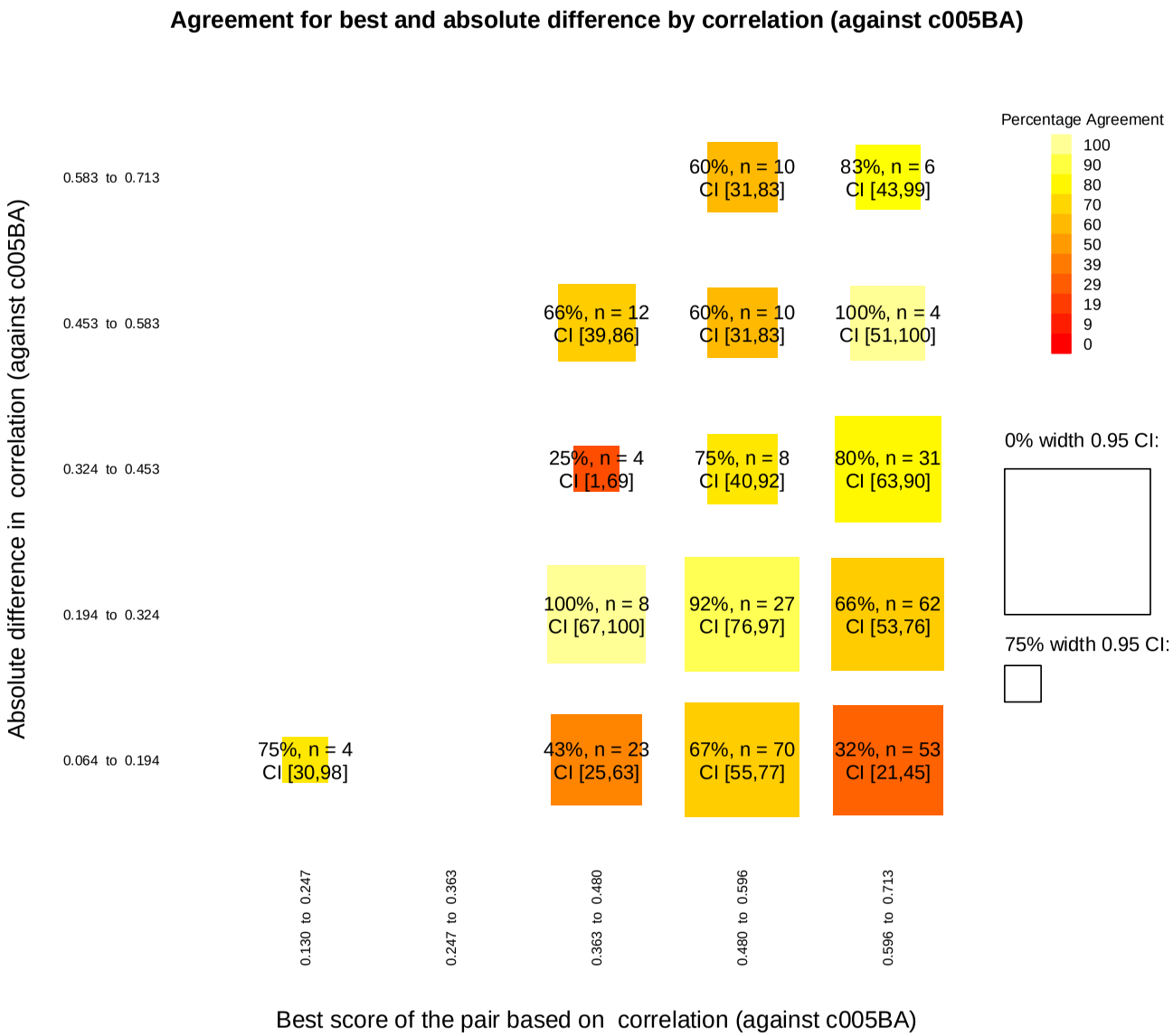


Figure 33 – Agreement with human evaluators of the C measure, binned by best score on the x-axis and absolute difference in scores on the y-axis.

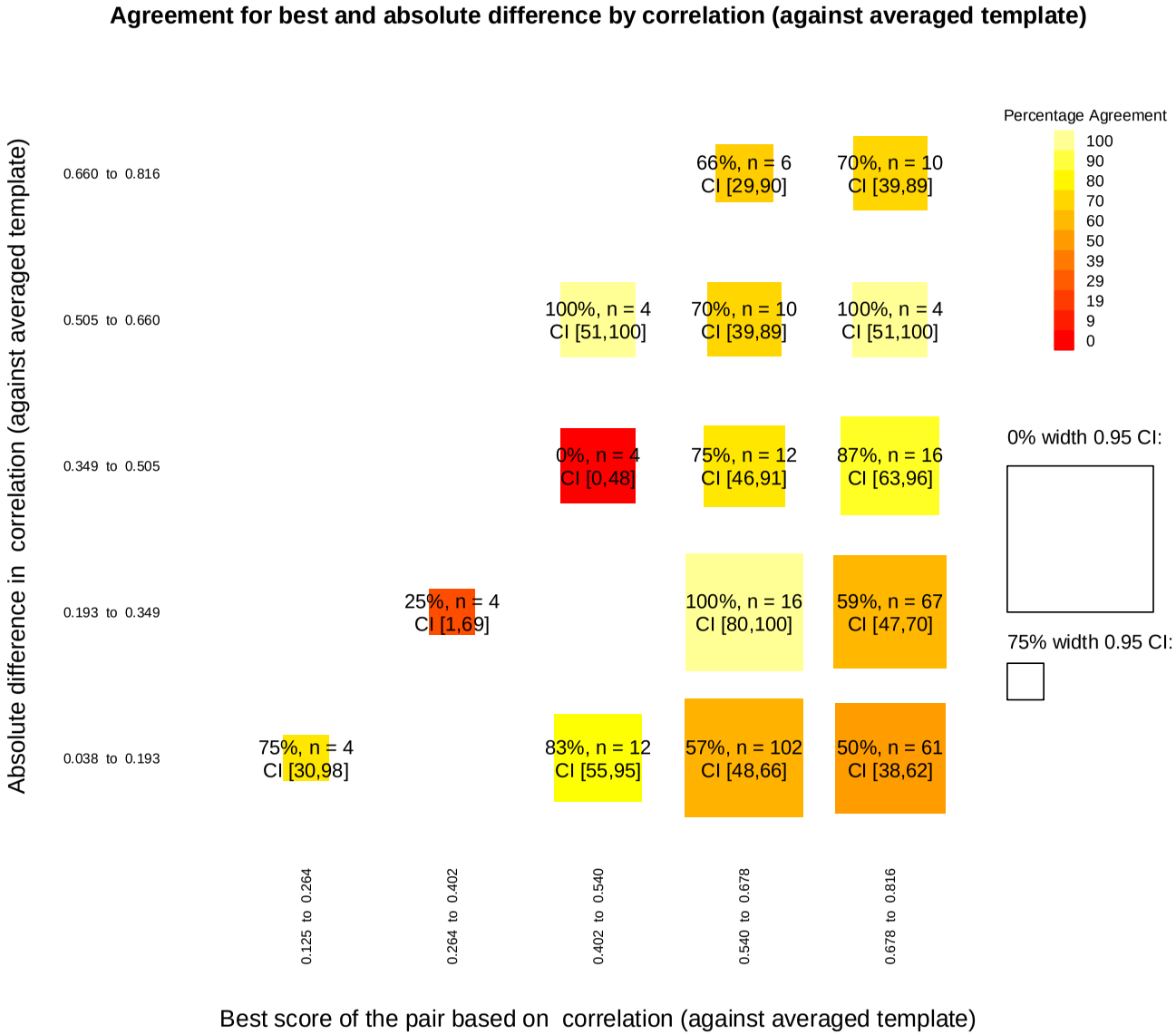


Figure 34 – Agreement with human evaluators of the C-S measure, binned by best score on the x-axis and absolute difference in scores on the y-axis.

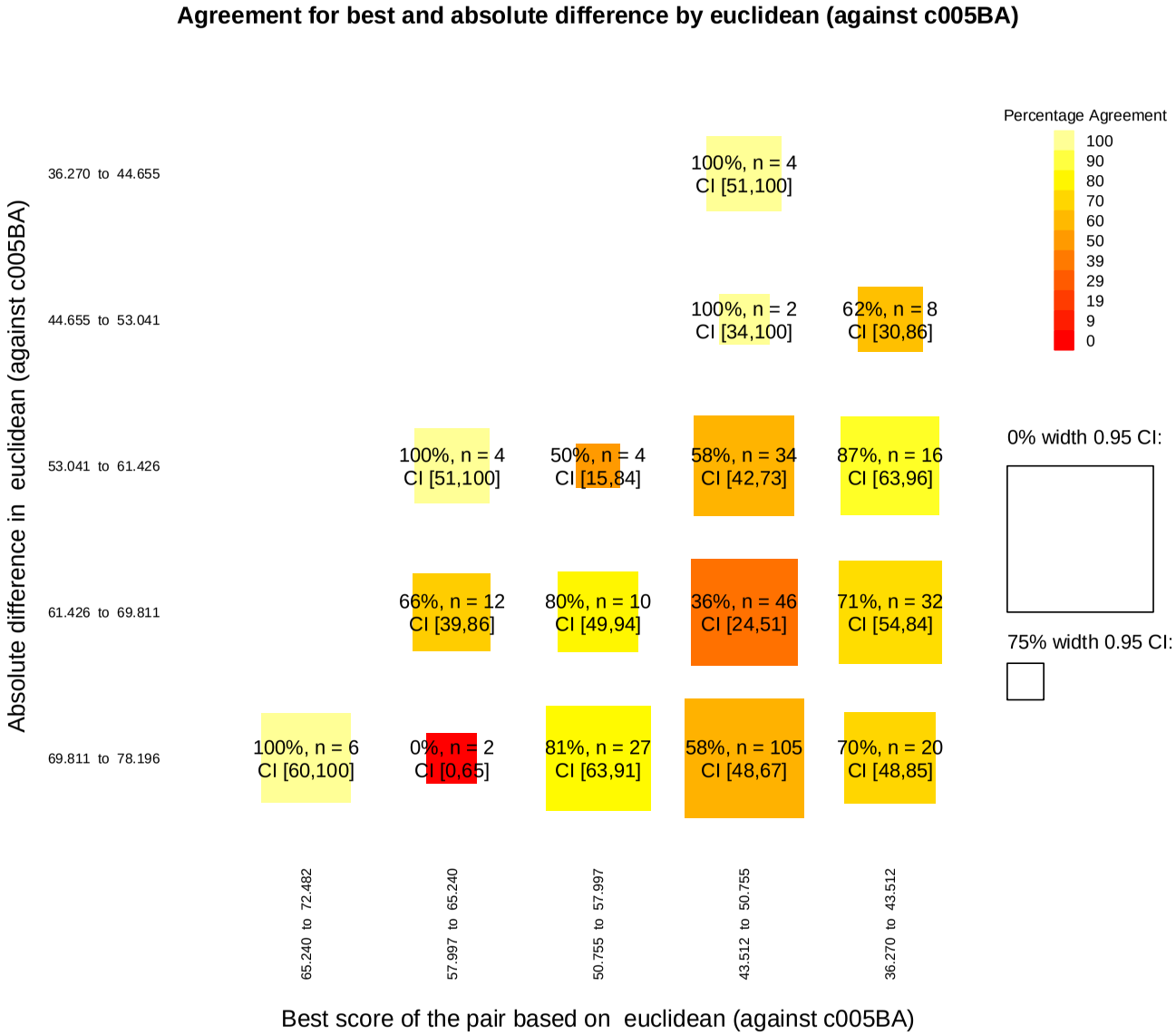


Figure 35 – Agreement with human evaluators of the E measure, binned by best score on the x-axis and absolute difference in scores on the y-axis.

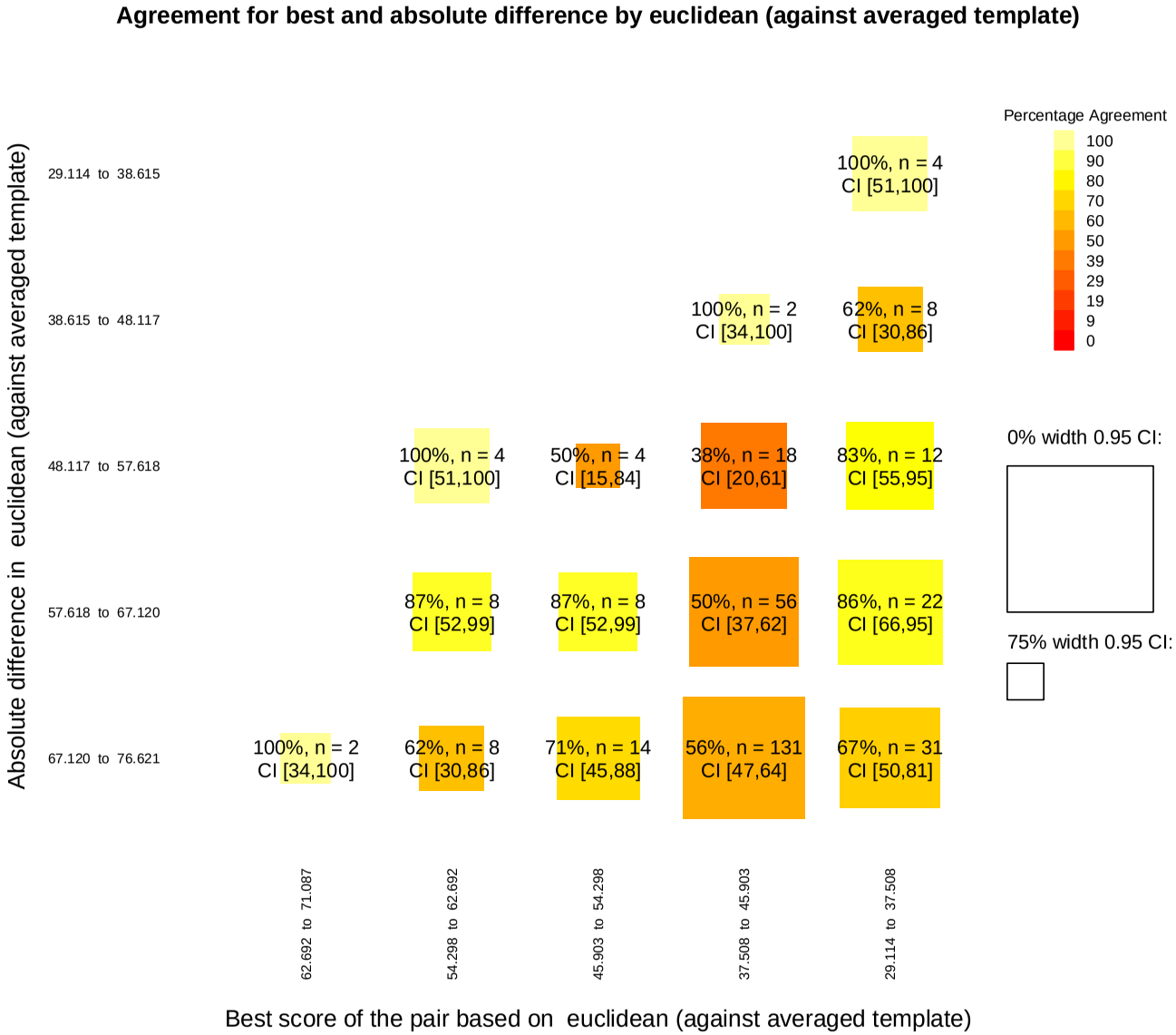


Figure 36 – Agreement with human evaluators of the E-S measure, binned by best score on the x-axis and absolute difference in scores on the y-axis.

are those based on mutual information (Figures 31 and 32). In both of these cases it is clear that a large absolute difference in scores corresponds to better agreement with human evaluators. Notably the agreement is particularly poor (worse than chance, in fact) where both scans are very good. A similar but less pronounced effect can be seen in the other measures. Unfortunately, the confidence intervals for most of these values are rather large relative to the differences in values, so I should caution against overinterpretation of these graphs.

A similar way of looking at the same data is to bin the pairs according to the range of the better score of the pair on the x-axis and the worse score of the pair on the y-axis. These data are shown in Figures 37 to 42.

Again, all the measures in this case broadly show the kind of distribution we would expect - there is higher agreement in the bottom right corner of each graph (indicating a high difference between the pairs of registrations) than along the diagonal where the scores are very similar.

A more traditional presentation of the relationship between agreement and absolute difference between the scores (i.e. the binning method above that shows the greatest effect) is shown in Figures 43 to 48. While showing one fewer dimension of data, these have the advantage that the error bars (showing 95% confidence intervals) are comparable to the values on the axes. From these graphs we can say, for instance, that with absolute differences of mutual information above 0.186 bits we see levels of agreement between the human evaluators and the mutual information measure which are significantly better than chance in every bin apart from a few with low numbers of observations.

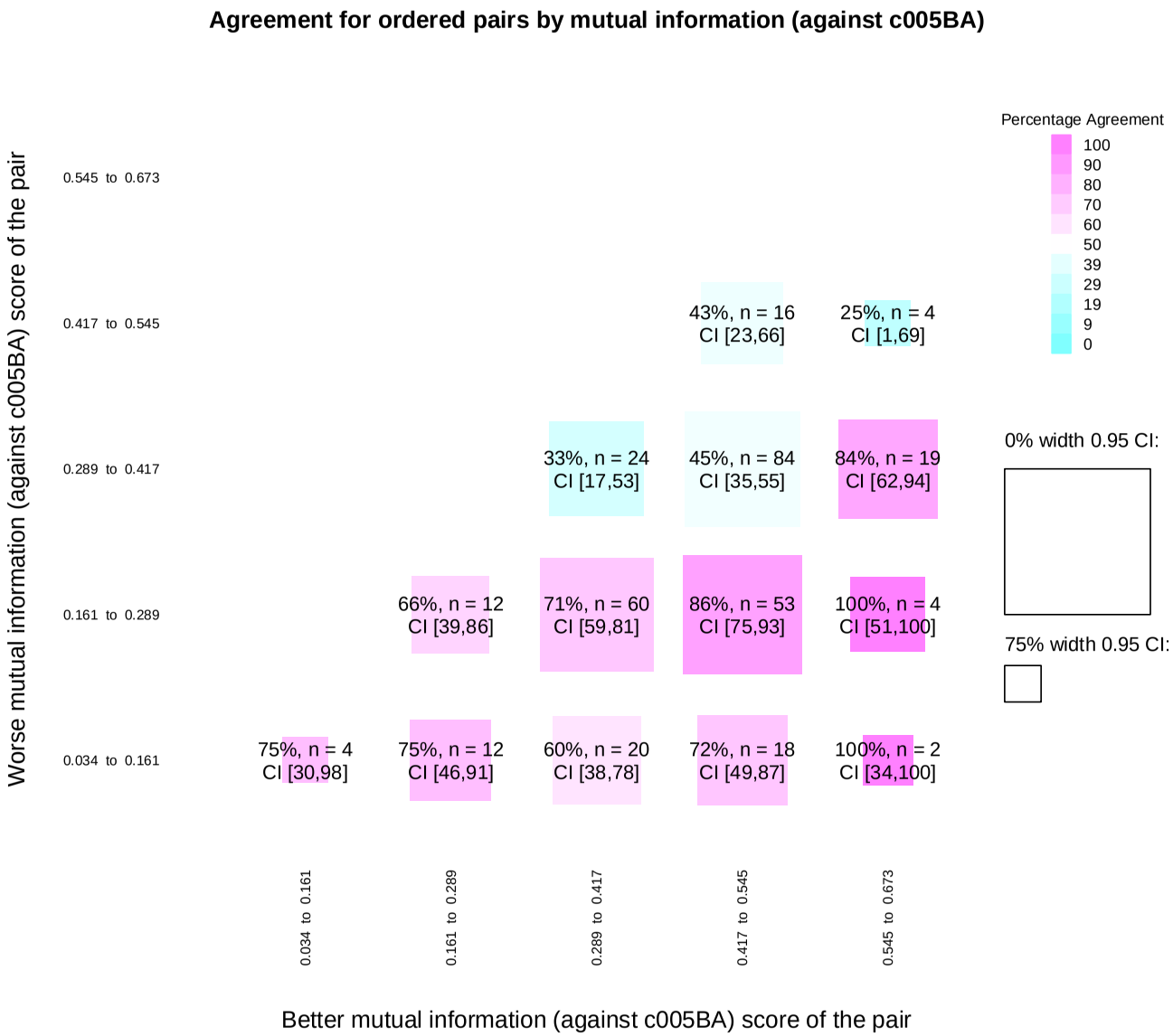


Figure 37 – Agreement with the MI measure, binned according to the better score of the pair on the x-axis and the worse score of the pair on the y-axis.

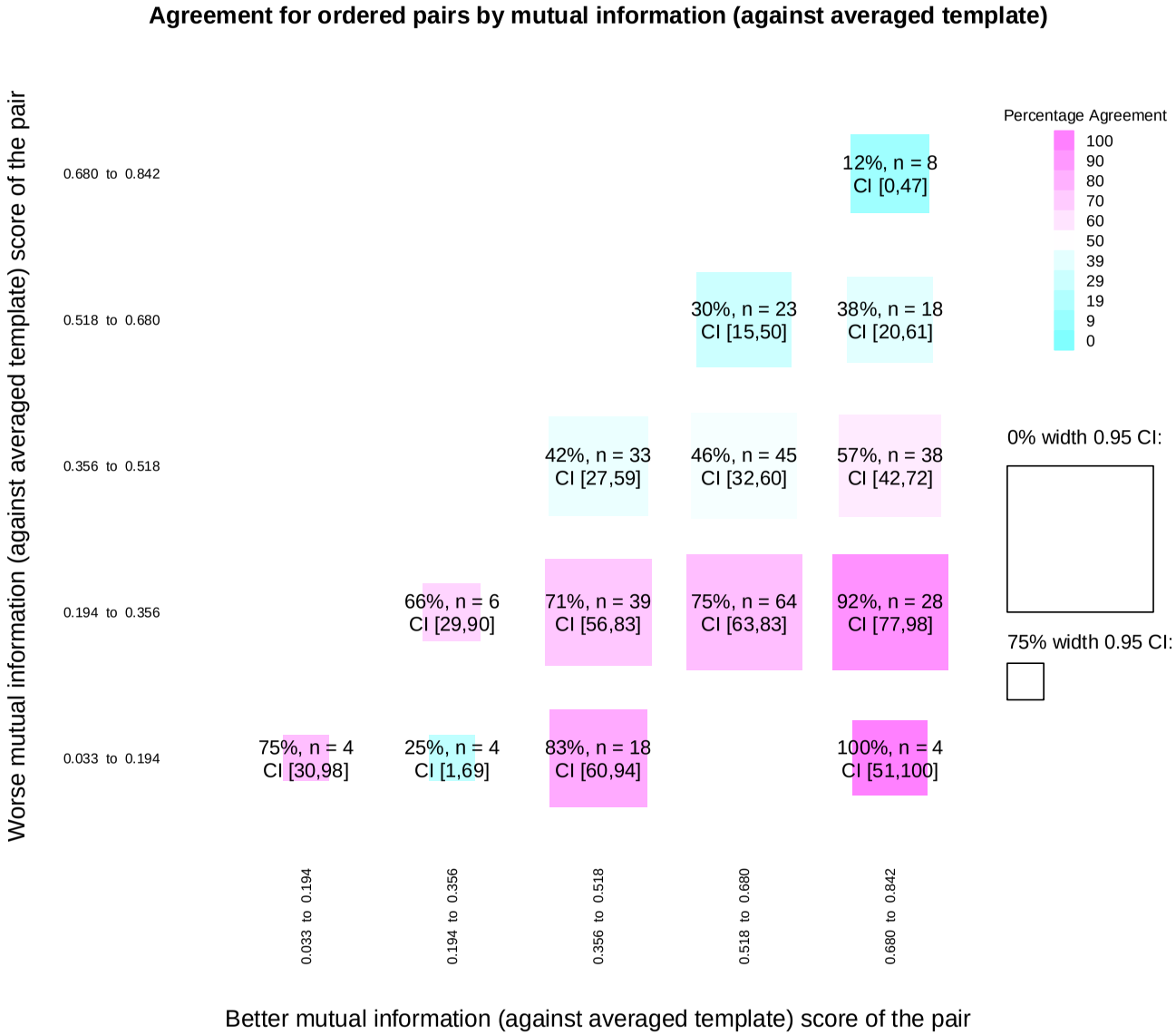


Figure 38 – Agreement with the MI-S measure, binned according to the better score of the pair on the x-axis and the worse score of the pair on the y-axis.

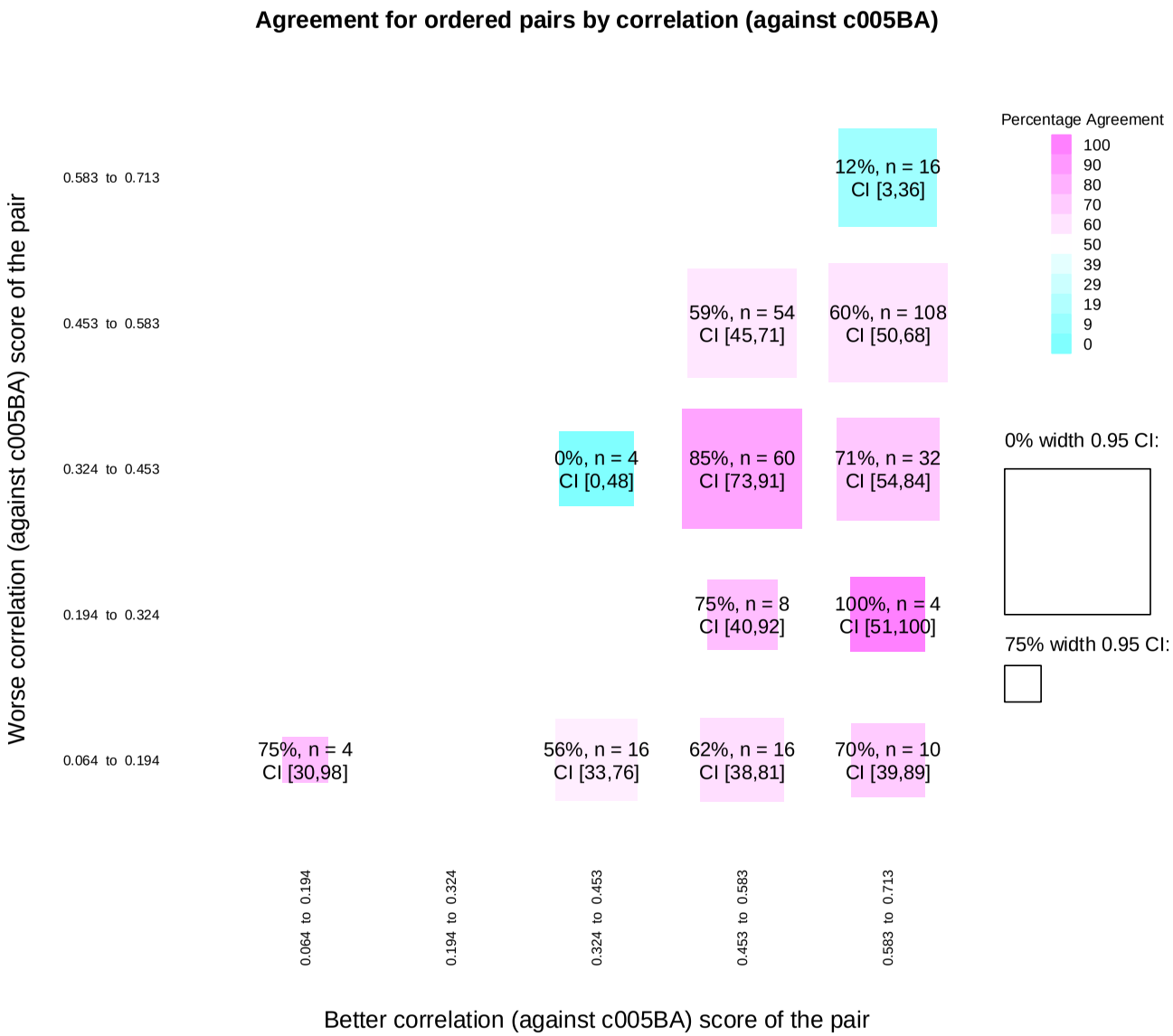


Figure 39 – Agreement with the C measure, binned according to the better score of the pair on the x-axis and the worse score of the pair on the y-axis.

Figure 40 – Agreement with the MI-S measure, binned according to the better score of the pair on the x-axis and the worse score of the pair on the y-axis.

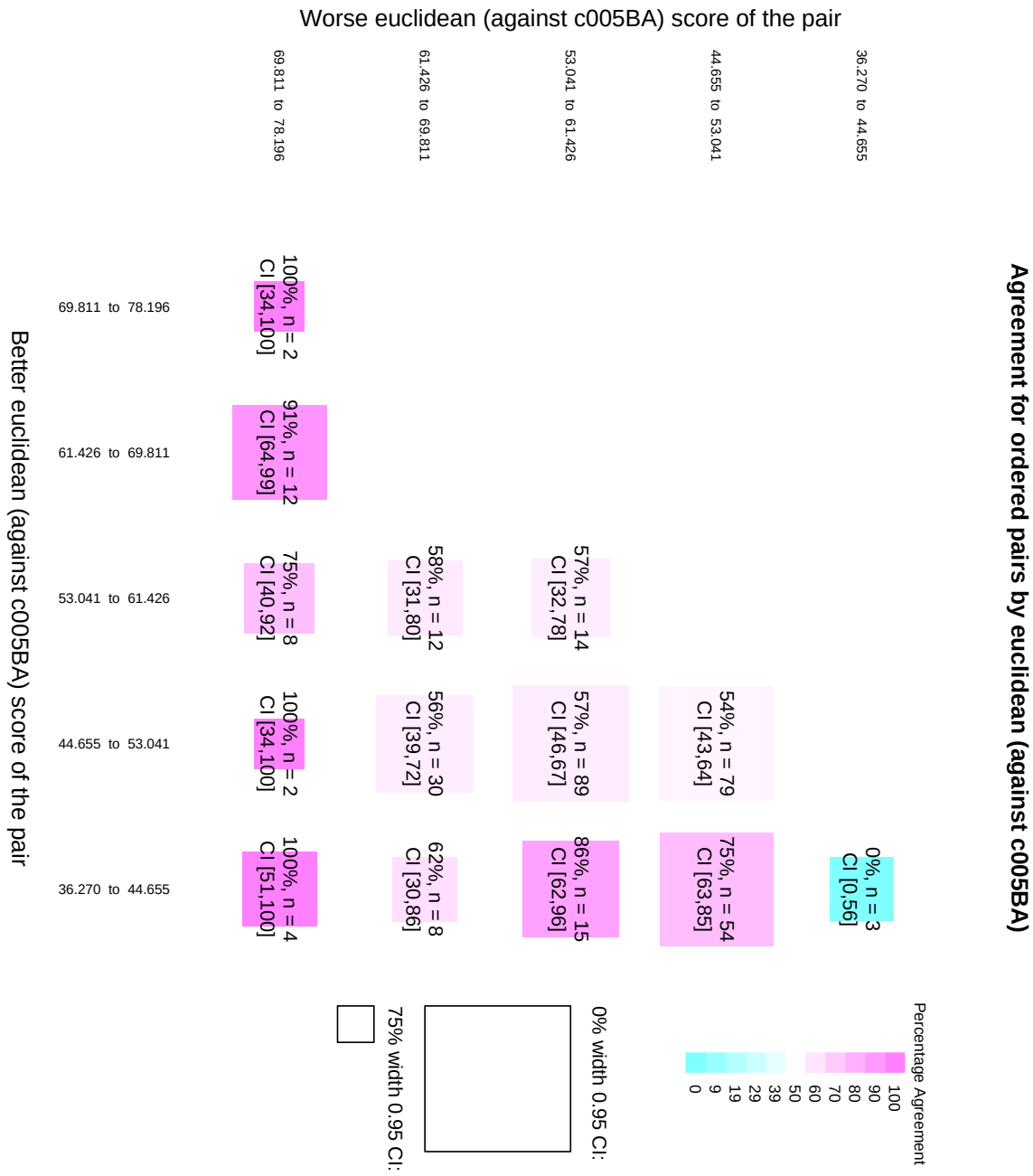


Figure 41 – Agreement with the E measure, binned according to the better score of the pair on the x-axis and the worse score of the pair on the y-axis.

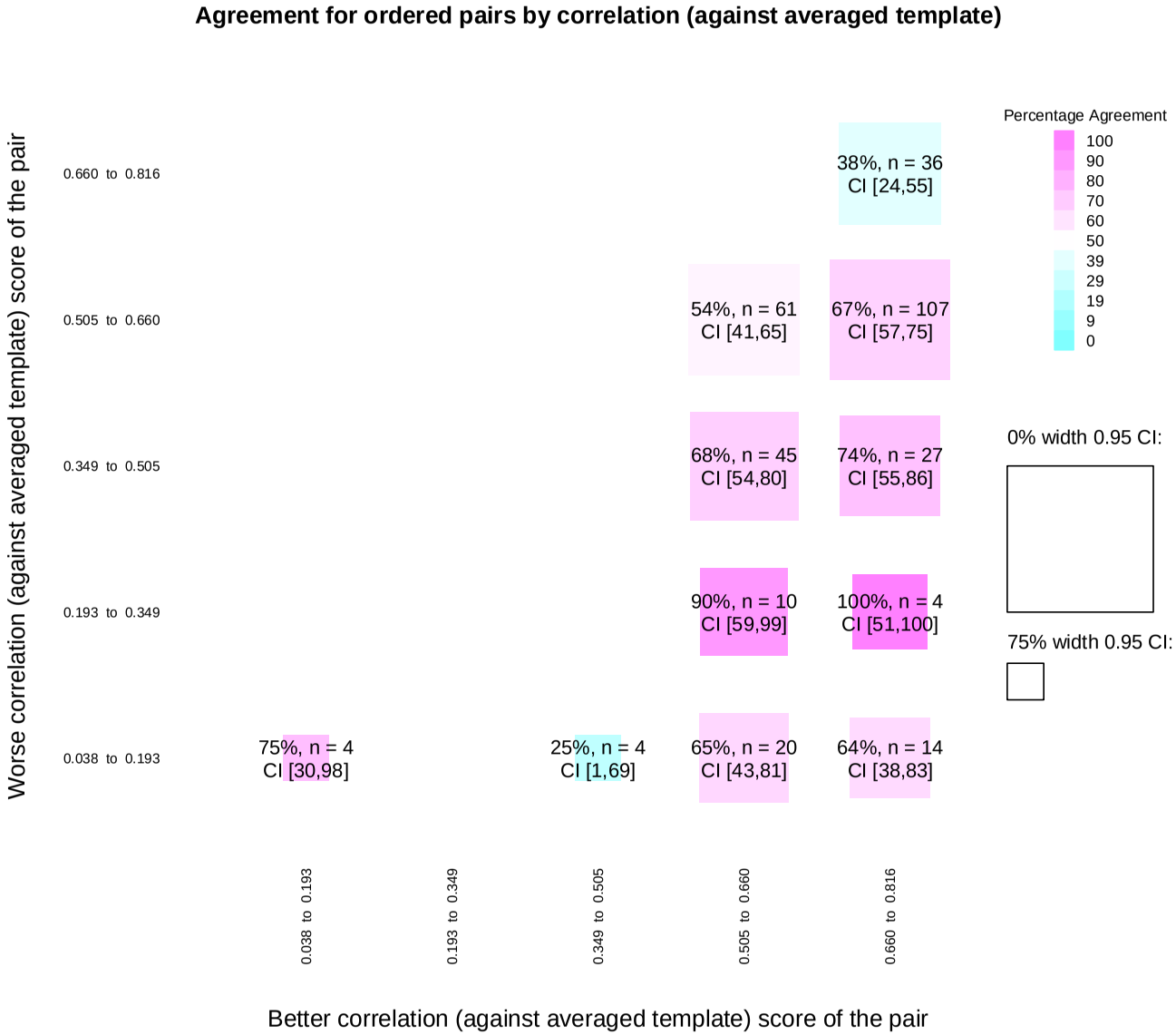


Figure 42 – Agreement with the MI-S measure, binned according to the better score of the pair on the x-axis and the worse score of the pair on the y-axis.

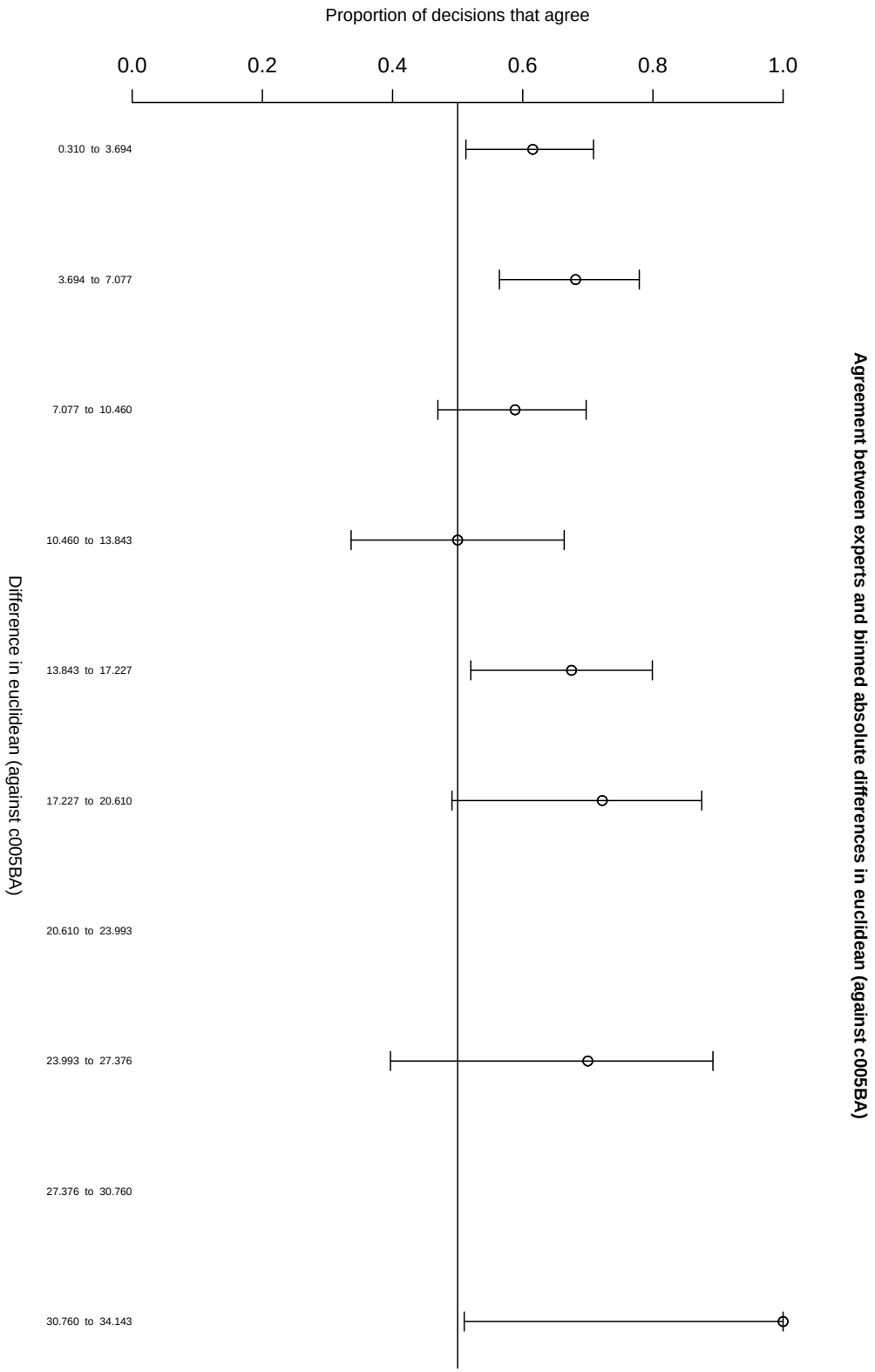


Figure 43 – Agreement between all the human evaluations and those of the automatic method E, binned by absolute difference in the score.

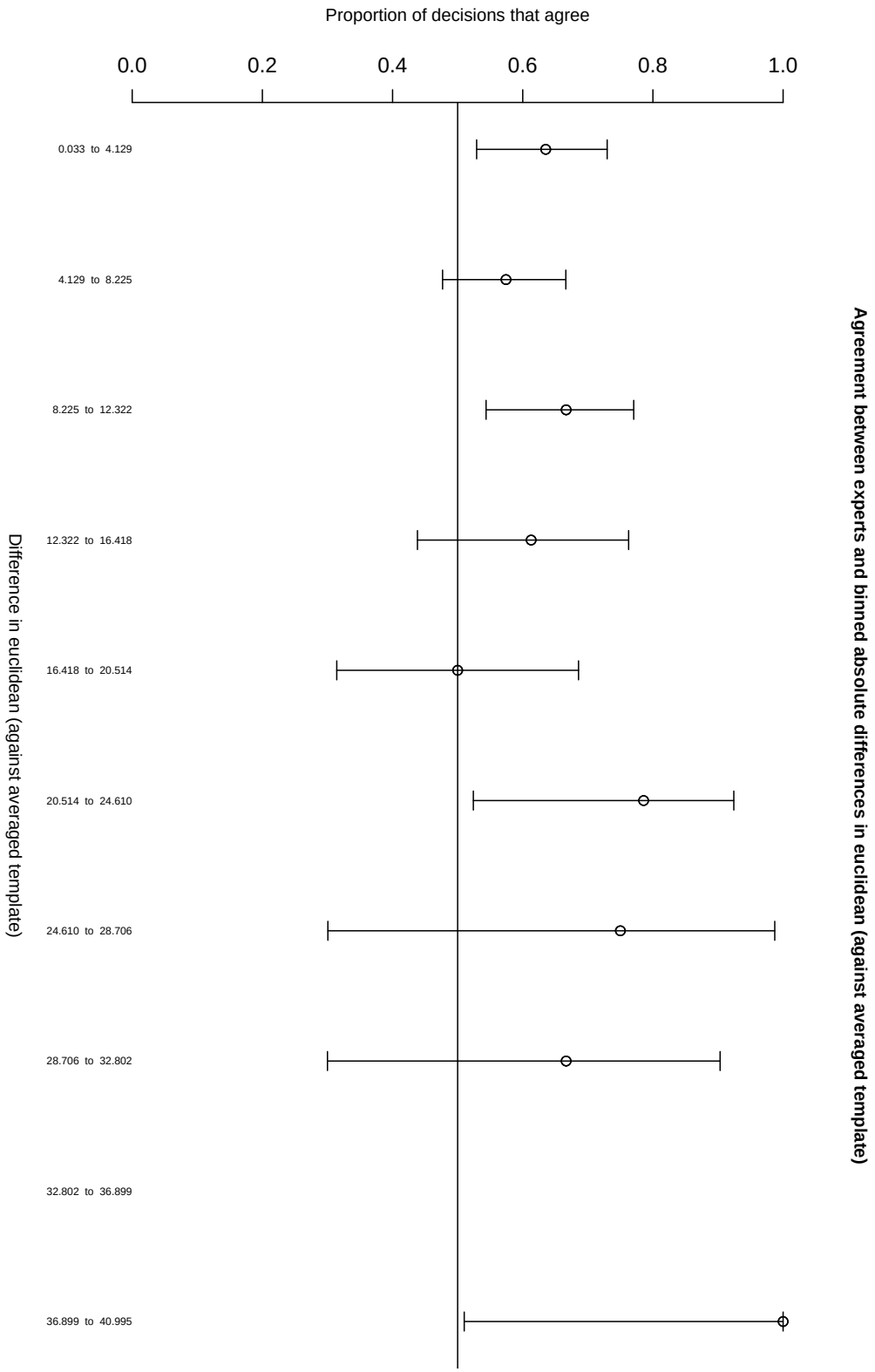


Figure 44 – Agreement between all the human evaluations and those of the automatic method E-S, binned by absolute difference in the score.

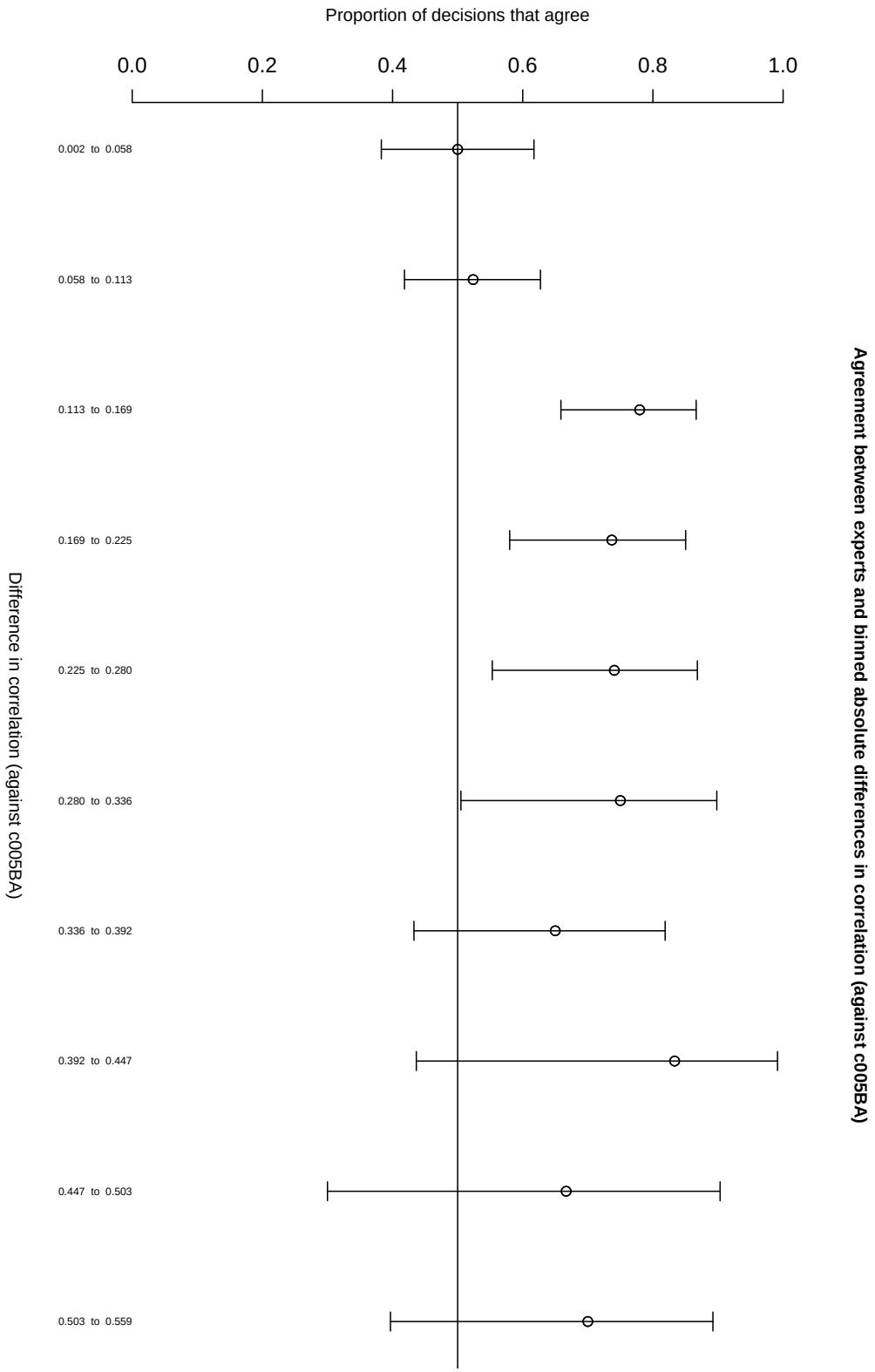


Figure 45 – Agreement between all the human evaluations and those of the automatic method C, binned by absolute difference in the score.

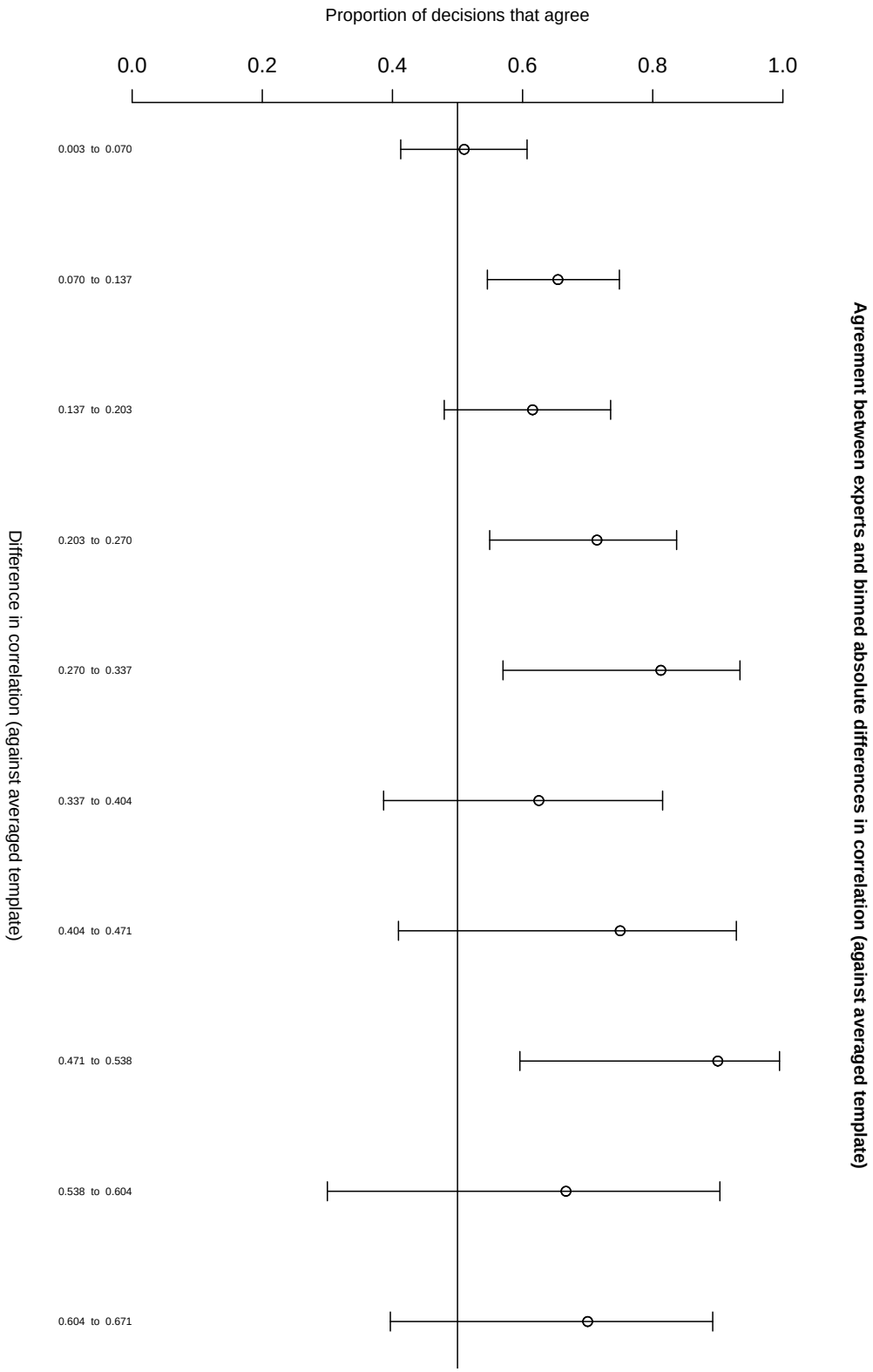


Figure 46 – Agreement between all the human evaluations and those of the automatic method C-S, binned by absolute difference in the score.

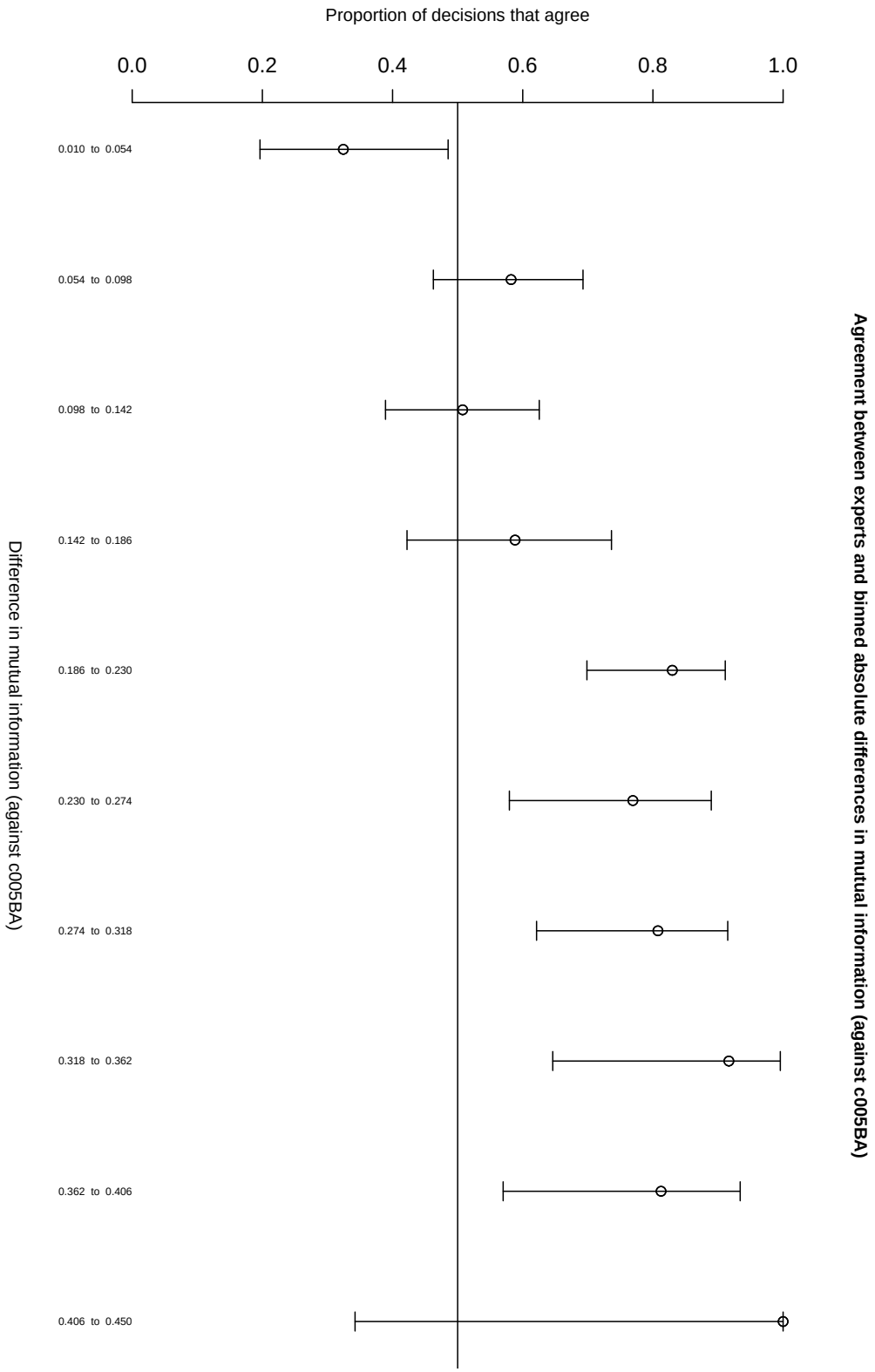


Figure 47 – Agreement between all the human evaluations and those of the automatic method MI, binned by absolute difference in the score.

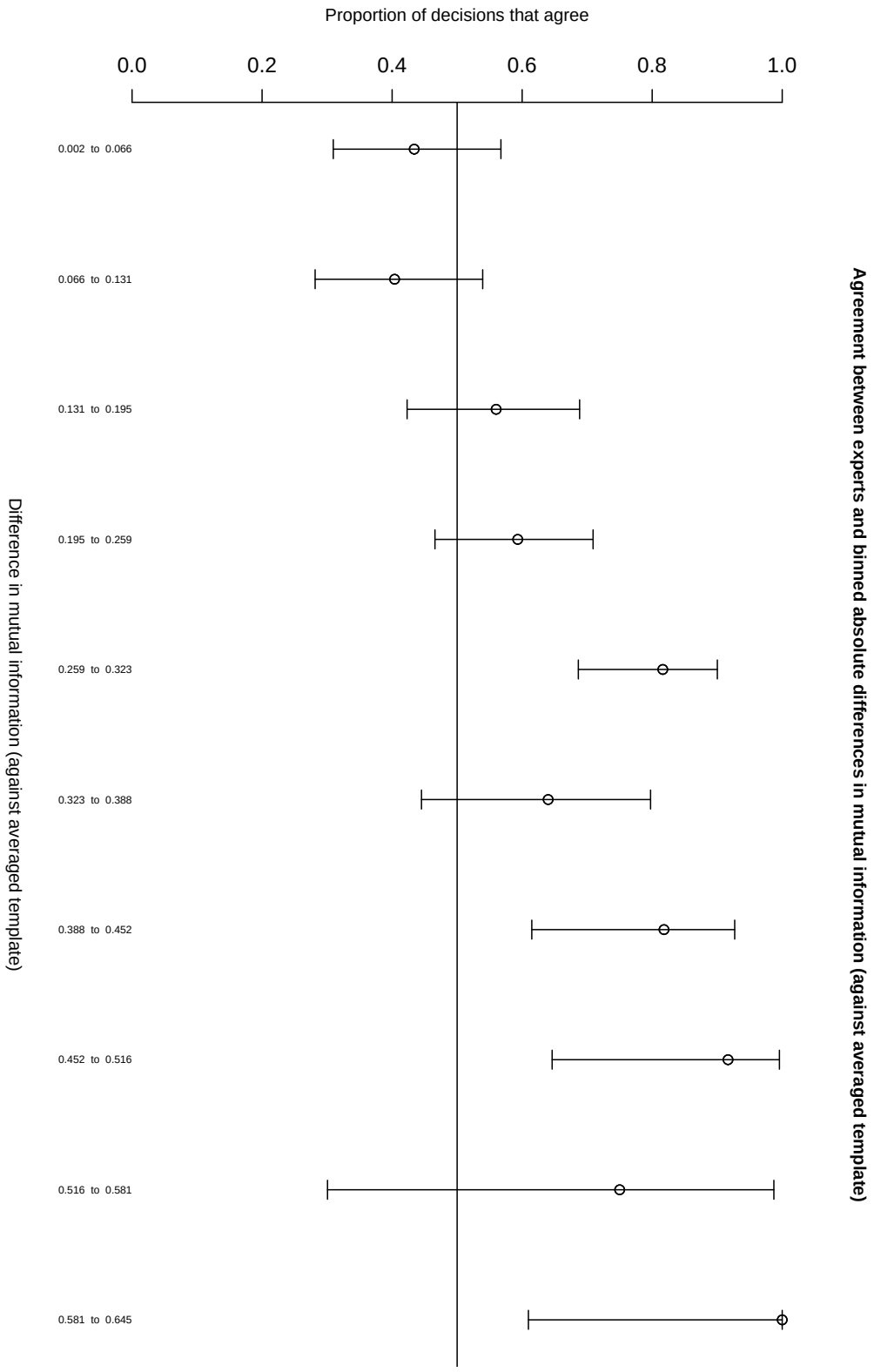


Figure 48 – Agreement between all the human evaluations and those of the automatic method MI-S, binned by absolute difference in the score.

4.7.5 Summary of Results

There are a few important problems with the analysis above which should be made clear:

- The number of human experts and the number of trials I was able to persuade them to do were unfortunately much lower than we would like, and this is reflected in the disappointingly broad confidence intervals in some of these graphs.
- The experts were asked to make their evaluations based largely on the central complex, but were still presented with the whole of both image volumes. Since we were scoring on a restricted region of interest it would have been better to restrict the pairwise comparisons by the experts to this same region.
- The nc82 channel of images in the corpus is sometimes very noisy. Since this is a more tricky antibody to use effectively than anti-β-galactosidase or anti-GFP, some images were included despite the noise, if the other channel was still good. A couple of the experts indicated that with a few of these scans the noise made it difficult to make an assessment.

Nonetheless, the data presented above do broadly support the assumption that these measures correspond with human assessments, particularly in case of mutual information, where the improvement in agreement with greater difference in scores is most similar to the distribution we would intuitively expect. These data also highlight that even within a group of experienced human evaluators there can be rather low levels of agreement.

I believe that this kind of analysis is important to conduct when evaluating registration algorithms, particularly in cases such as this where we have an atypical set of images. In the future it would be excellent if we could run a similar but larger experiment with more experts.

4.8 Registration Results

4.8.1 Results of Registration Method Comparisons

Since in the previous section we saw no clear evidence to suggest rejecting any one of the measures we considered, I present comparisons here of the different registration techniques according to all of the measures but only against the c005BA template. (The averaged template is actually generated

by one of these methods so obviously using that template for the evaluation would seriously bias those results.) Another reason to continue including results from the Euclidean and correlation measures is that the **cmtk** method specifically optimizes normalized mutual information, so using the mutual information measure alone may obscure a bias towards that technique.

In each of the following graphs (Figures 49 to 51) have ordered the images' filenames on the x-axis by the best score achieved for it by any of the methods under consideration. This has the effect that the images that can be most effectively registered (and thus which tend to have the highest quality nc82 channels) are consistently on the right hand side of of the graphs.

The results in these graphs are clear: the **cmtk** method produces consistently better results than the other methods except in two cases where the initial affine registration (performed by method **affineo**) fails.

One surprise is how poorly the **vib** method performs on this data set. Looking at the registered images, there are a couple of clear reasons why this might be the case. Firstly, the labelled ellipsoid body is a highly symmetrical structure, and the `RigidRegistration` module of VIB does not penalize registrations where it has, for instance, been rotated through 180°. The diffusion stage then creates a "whirlpool" warping around the ellipsoid body which renders the registration unusable. Secondly, the morphology of the protocerebral bridge makes it a particularly difficult neuropil region to effectively register: because of its thin and curved shape, if the initial guessed transformation is off then there may be no overlap at all and the optimization algorithm can take no small step to improve it. In contrast the registrations of the large and rotationally asymmetric fan-shaped body tended to be good.

I should make it clear that the results for the VIB protocol presented here are rather unfair to it in a number of respects. When the protocol was developed, much care was taken to choose a minimal set of brain spanning neuropil regions that would be most easy to register while not too being too time-consuming to mark up. In this trial we have simply taken the most obvious neuropil regions in the restricted region of interest that we are considering. Secondly, the ImageJ version of the VIB protocol uses a different optimization method for rigid registration from the Amira version, and we have observed a number of problems with it.[39] It is likely that the built-in optimization in Amira performs more effectively.

[39]This is `ConjugateDirectionSearch` from the `pal` library, which is based on work by Prof Richard P. Brent in 1973. Prof Brent himself suggests on his web page considering more modern research into numerical optimization.

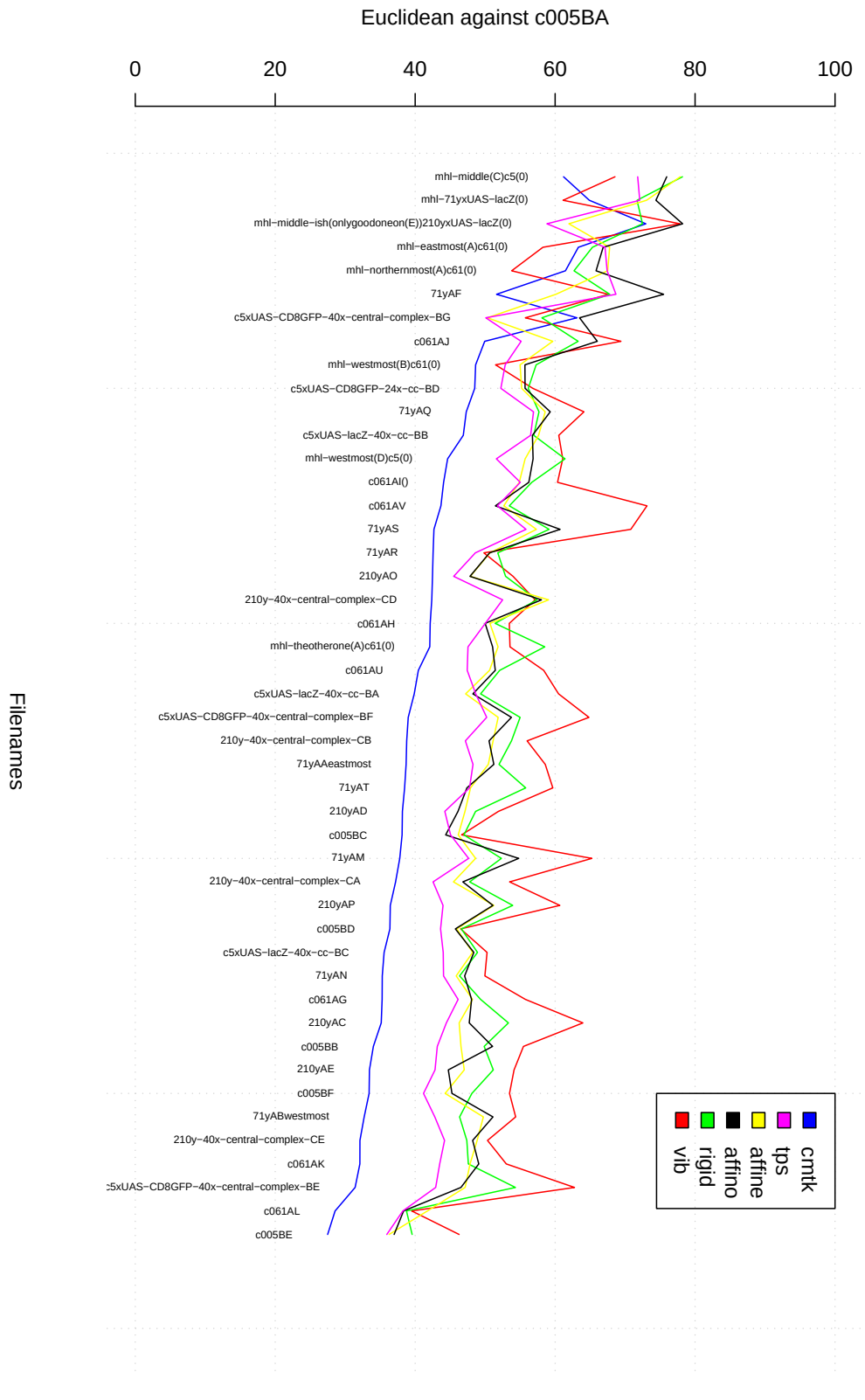


Figure 49 – Evaluation of registration algorithms according to the Euclidean measure against the c005BA template. (Lower values of the measure mean the registration is better.) The abbreviated names for the methods are explained in section 4.4.

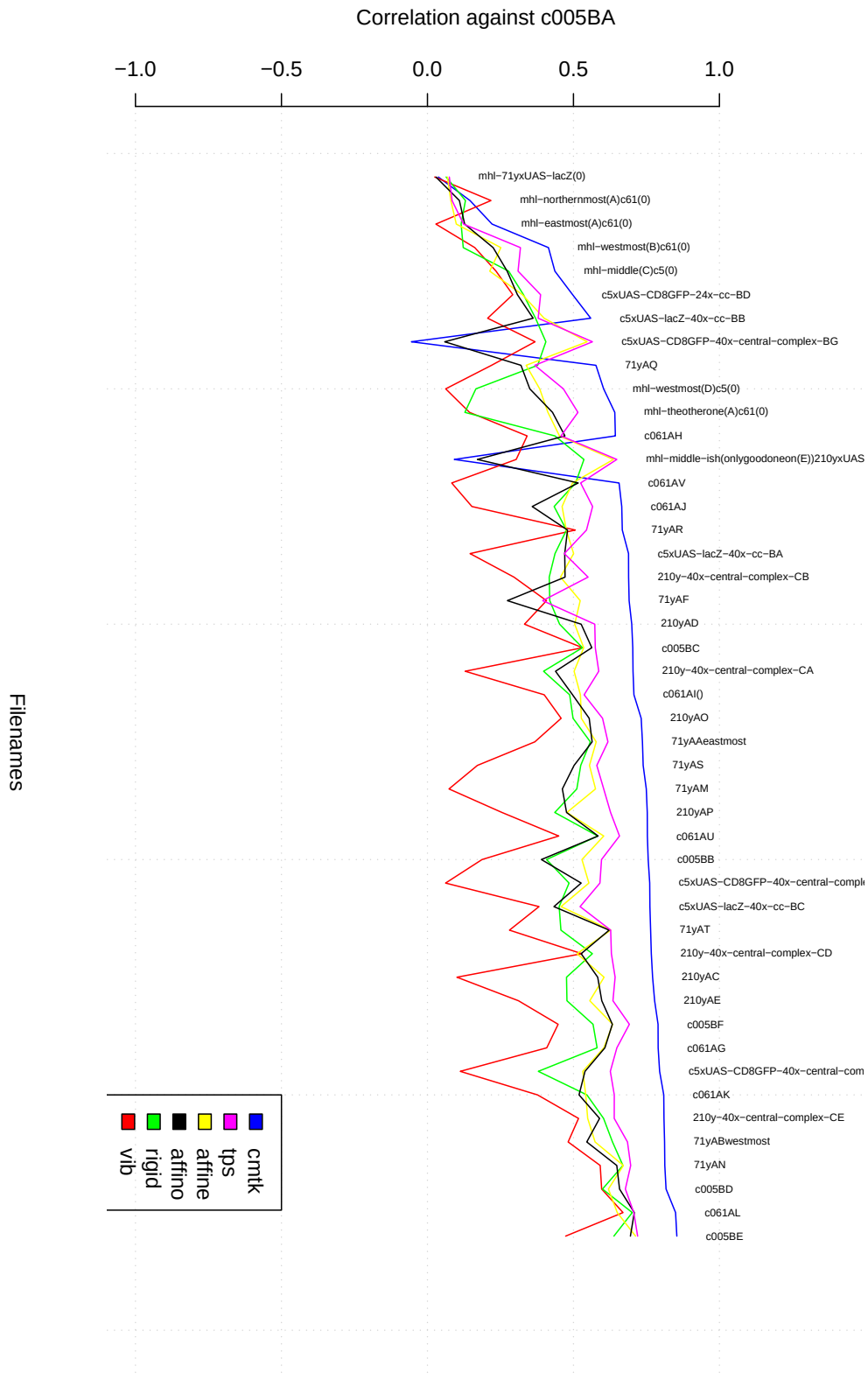


Figure 50 – Evaluation of registration algorithms according to the correlation measure against the c005BA template. (Higher values of the measure mean the registration is better.) The abbreviated names for the methods are explained in section 4.4.

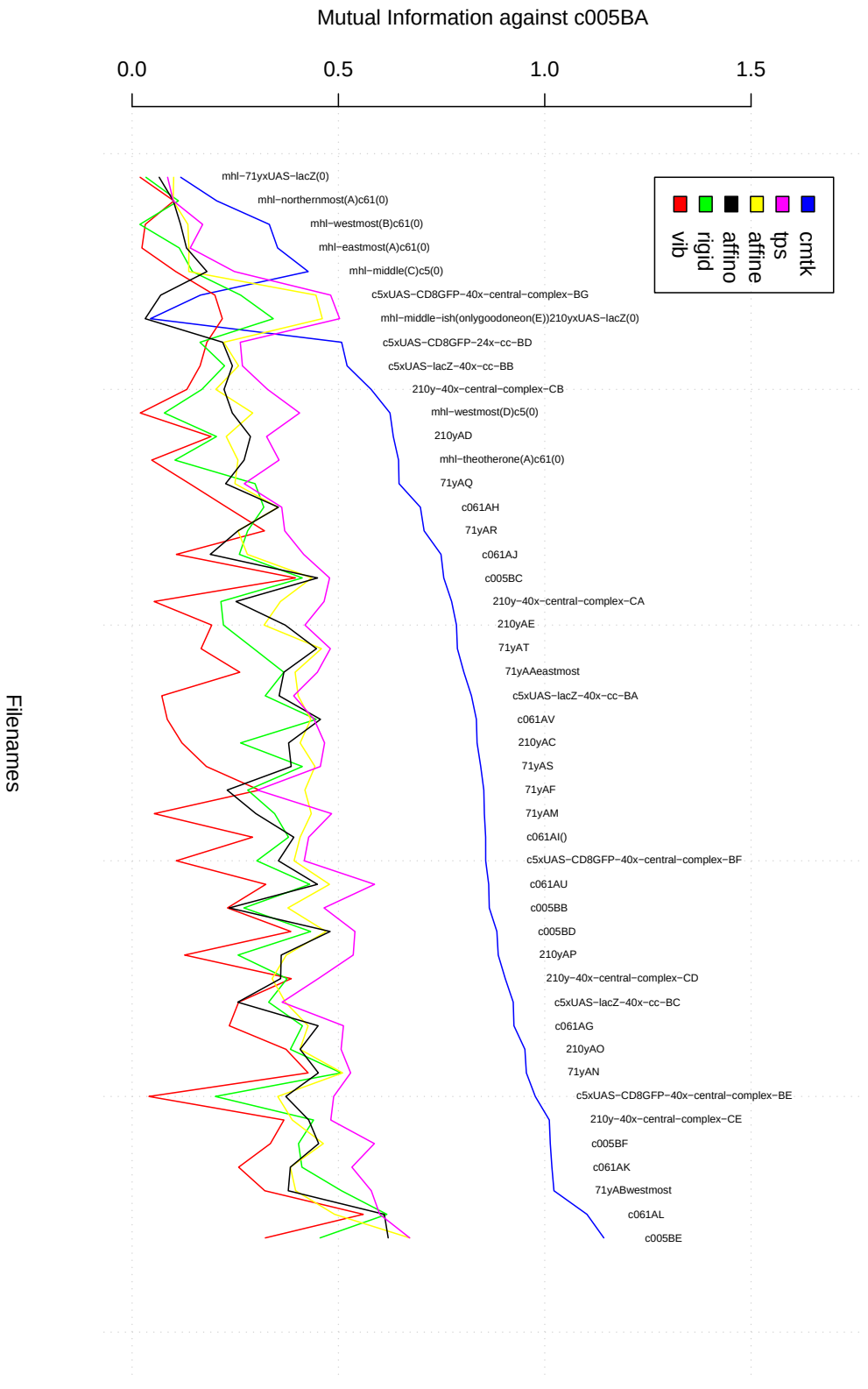


Figure 51 – Evaluation of registration algorithms according to the mutual information measure against the c005BA template. (Higher values of the measure mean the registration is better.) The abbreviated names for the methods are explained in section 4.4.

While the Computational Morphometry Toolkit clearly is most effective, it is worth pointing out that pragmatically the second best method (**tps**) may often be a better choice. An often-overlooked aspect of registration is that one doesn't always need the very best possible registrations for the task at hand, and in many cases the **tps** results are quite good enough - given that this method takes a matter of minutes to produce a registration, as opposed to an hour for CMTK on typical desktop hardware, I suggest that the second conclusion of this chapter is that the thin plate spline method from manually chosen landmarks is a good choice for fast, and often sufficient, registrations.

If we discount the poor performance of the VIB protocol on this data set, which is explained above, another important point is that the other non-linear registration algorithms outperform the linear ones. This may not be particularly surprising, but is useful to establish for these central brain images, since, as mentioned above, they do not include the problematic optic lobes, which are often mentioned anecdotally as one of the reasons that elastic registrations algorithms are necessary for the fly brain.

An expected criticism of the type of survey done in this chapter is that there are promising techniques that have been omitted from the survey. In response to this, it is worth reiterating that the literature on non-linear registration techniques is vast, and freely available implementations of the techniques described therein are not always available. We have been guided in our choices both by pragmatic considerations (e.g. that it would not have been a good use of time to implement large numbers of complex registration techniques) and by published reports of these techniques working well on insect brain images.

The following subsections discuss some possible further work on registration that I have not had enough time to complete. Although some of these should be trivial in implementation terms, rerunning the evaluation with new variant methods can take the better part of a week of computer time.

4.8.2 Refining Manually Picked Landmark Points

There are two obvious problems with the manual selection of landmark points as I have done for the basis of the registration in this chapter:

1. Picking points precisely in 3D from the slice-by-slice view used by ImageJ is difficult to do with precision. This is particularly the case if the brain is at an odd angle, or if the target

location is difficult to find. For example, we use the top of the alpha lobes in this study, which is a smooth surface - the exact top of this region is tricky to find precisely.

2. Different annotators may interpret the descriptions of the landmarks in different ways.[40]

3. Some of the neuroanatomical features we are interested in may be very small in these images - only a couple of voxels wide. For several of the methods, this means one must consistently place the landmarks at least as accurately as that.

One solution that potentially fixes all three of these problems is to use a very localized registration to fine-tune the placement of the points.[41] This works effectively in 2D stitching applications such as Hugin[42] for refining the manual placement of landmark points. I added some experimental functionality of this kind to the `Name_Landmarks_and_Register` plugin,[43] which is available when a template image is loaded. Once a landmark point has been manually placed, if the "fine-tune" option is selected, then the following happens:

1. A small region around the landmark in the model image and a similar sized one in the template image are cropped out.

2. A set of 25 initial transformations is calculated from:

   (a) The best rigid transformation (as used in the **rigid** method above) between the landmark points.

   (b) The 24 distinct rigid mappings which map one axis onto another.

3. Multiple threads ($n+1$, where n is the number of processors on the system) are started, each of which uses a different initial transformation and attempts to optimize the rigid transformation using the Conjugate Direction Search method, scoring the registrations with correlation.[44]

[40] This was not a practical problem in this work, since I was the only annotator, but may be an issue for collaborative work.

[41] An alternative solution to just the first of these problems is to use the 3D viewer for picking points. I may add this feature in the future.

[42] `http://hugin.sourceforge.net/`

[43] The code to enable this feature is commented out from the released version of `Name_Landmarks_and_Register` in Fiji, but can be re-enabled by setting the boolean field `offerFineTuning` to `true` in `Name_Points.java`.

[44] Correlation was chosen rather than mutual information, since in these small regions it may be that the inverse of the image matches well - this may produce a high score with MI, but will not with correlation.

4. Each time a thread finishes, optimization from the next initial transformation is started until there are no more initial transformations to be tried.

5. A final optimization is attempted, starting from the best transformation found so far in the previous steps.

The user interface for this is shown in Figure 52.

The results of using this method are interesting, but not yet good enough for the fine-tuned landmark points to provide better registrations than the manually placed ones. One point particularly worth noting from this trial is that even though a small region of the image is cropped out for the registration, for some landmarks the registration is still dominated by unexpected features in the surroundings. For example, registration of the most superior points of the protocerebral bridge tend to be somewhat off, while certain surrounding neuropil regions (the "antlers" in the new nomenclature) will be exactly registered. Interestingly, this is not necessarily a problem - we would expect, both from anecdote and the relative fuzziness of the protocerebral bridge in averaged templates, that the position of the antlers is more consistent than the protocerebral bridge in relation to the rest of the brain. In other regions, similar effects create a more serious problem. For example, at the top of the brain there are often variations in the shape of the rind around the top of the alpha lobes (possibly from dissection damage) and these typically dominate the registration scoring over the protrusion from the mushroom body.

Currently there appear to be too many misregistrations during the fine-tuning for it to be effective to use this method across the board, but it is promising enough to pursue further in the future. In particular, I have not yet compared the scores from good and bad registrations - it may be that by setting a high cut-off we can simply filter out the misregistrations and keep the good refinements.

4.8.3 Improving Initial Affine Registration for CMTK

As mentioned in the description of the affine registration methods above, a simple hybrid approach between the **affine** and **affineo** techniques should improve the initial affine registrations, and hopefully will correspondingly improve the results from CMTK. At the very least we would hope that this will correct the two failure cases, since the former affine registration technique succeeds where the latter fails.

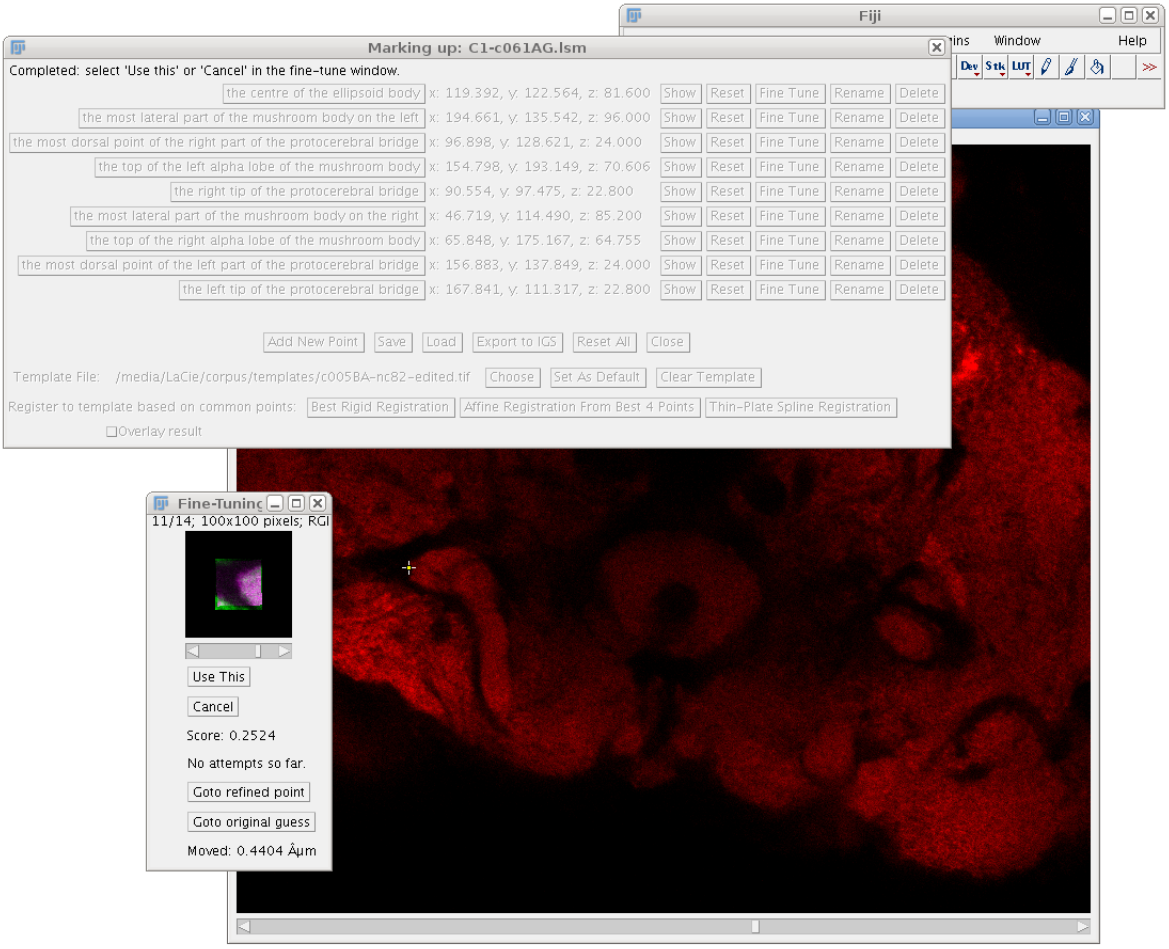


Figure 52 – A screenshot of the "fine-tuning" interface in Name_Landmarks_and_Register. The small window shows the progress of fitting a small region around the landmark in the template and the model; if the result looks right, then the user should click "Use This" to move the landmark point accordingly to the equivalent point in the template.

4.8.4 Automatically Picking Landmark Points

As an alternative to manually picking landmarks, an interesting approach is to use an automatic landmark identification algorithm such as Scale Invariant Feature Transform (SIFT) [Lowe, 2004] or Multi-Scale Oriented Patches. [Brown et al., 2004] There is an implementation of the latter in the VIB repository, but so far the results of this implementation for fly brains have been poor. Nonetheless in the future I expect that methods such as these will supersede the manual selection of landmarks such as we have done for this chapter; the equivalent techniques for matching landmarks in 2D images for stitching and alignment work very well.

4.9 Generating the Averaged Template

I generated an averaged template brain by taking the best 30 registrations from the CMTK method according to mutual information and taking the mean value at each point. This produces a much smoother image than the original c005BA brain, as can be seen in Figure 53. One can clearly see that the structure of staves and layers in the fan-shaped body and details such as the inner and outer peduncles are preserved after this averaging, which is as we would hope and expect.

Figure 54 shows a slice through the kind of stack that one can generate by overlaying these registered images.

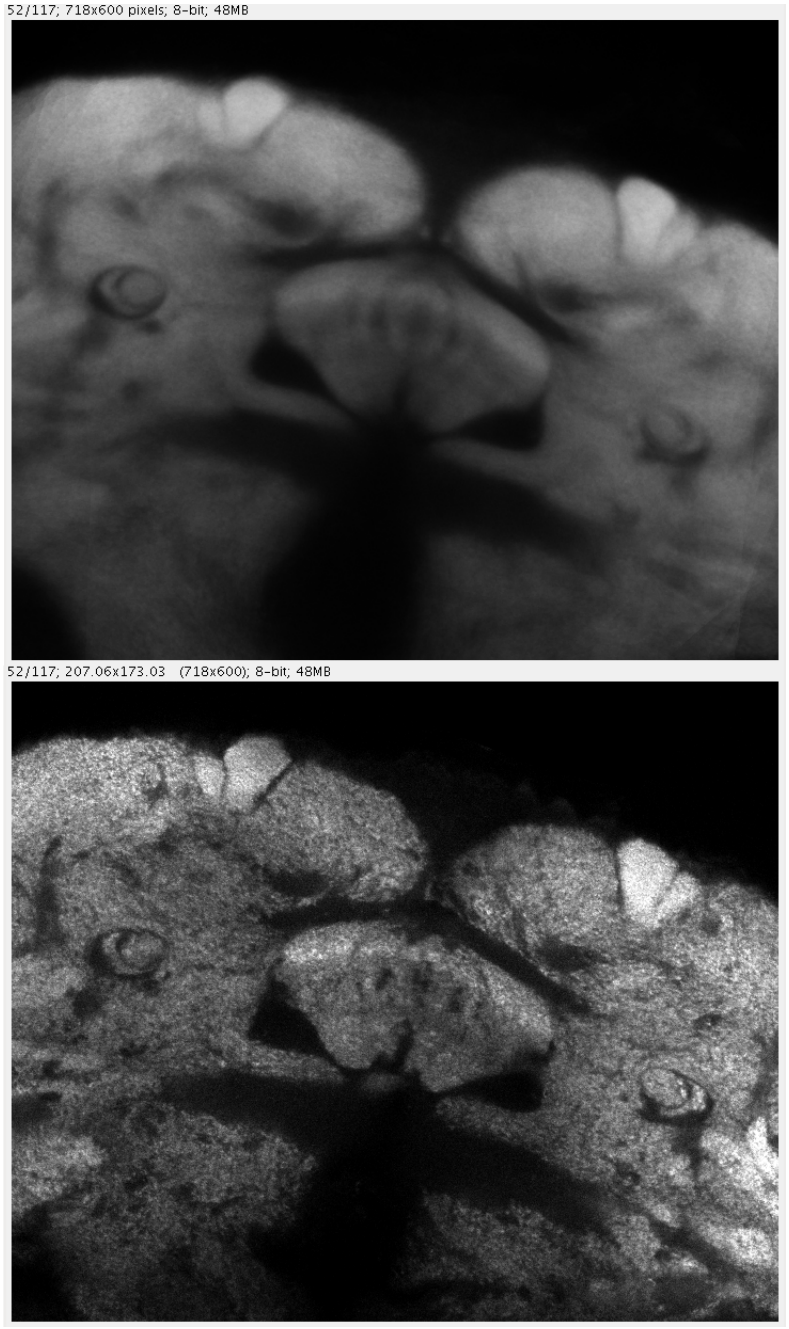


Figure 53 – These are a representative slice through the c005BA image (below) and a corresponding slice from an average of brains from the best 30 registrations by CMTK (above).

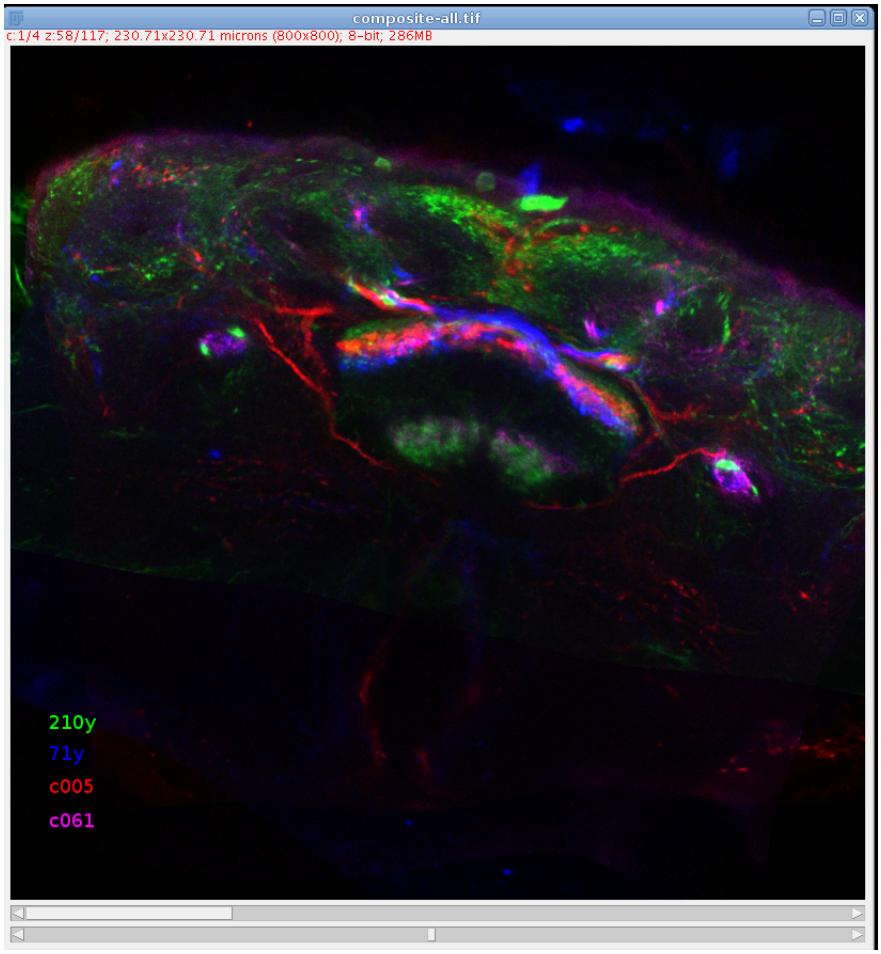


Figure 54 – This shows a slice through the fan-shaped body showing the GAL4 signal from the best four registered images, one from each line.

5 Connectivity

In this chapter, I describe the tools I have developed for accurately tracing neurons in image stacks with various levels of user intervention. I also present the results of applying these tools to the images of the *Drosophila* brain acquired in Chapter 2.

5.1 Preliminary Considerations

There are some important caveats which should be made clear in the first place about this work. The first of these is that although we have chosen GAL4 lines with relatively sparse expression patterns, they are still made up of overlapping fluorescence from hundreds of neurons in every scan. This makes it difficult to defend statements about connections between different regions based on this data. For example, an apparent connection may be composed of two neurons that either cross over or run parallel before diverging, to take two common examples. This problem is shown in Figure 55, which gives examples of different neuronal topologies that might produce a very similar image in the confocal stack.

Another common source of confusion is that neurons may well pass through a neuropil region without making any synaptic connection there. Although the conventional *Drosophila* neuropil regions are defined by the nc82 marker for active zones, and are thus in general rich in synapses, this does not imply that any particular neuron that passes through the region will synapse there. One would need to use pre-synaptic and post-synaptic markers to be sure of this, or, alternatively, it is sometimes possible to infer where the synaptic regions are from expression patterns typical of dendritic or axonal arbors.

There are nonetheless some situations where we can make a convincing case that the patterns are best explained by a direct connection between two regions. For example, we can pick out paths in these images that are strikingly similar to some described in [Hanesch et al., 1989] or other literature. The crucial point is that with source data such as this, there must be a step where a human expert curates the results at some stage. One way of avoiding such problems may be to use different genetic techniques to enable smaller sets of neurons to be imaged, such as Mosaic Analysis with a Repressible Cell Marker (MARCM) [Lee and Luo, 2001], FLP-out (as used, for example, in Wong et al. [2002])

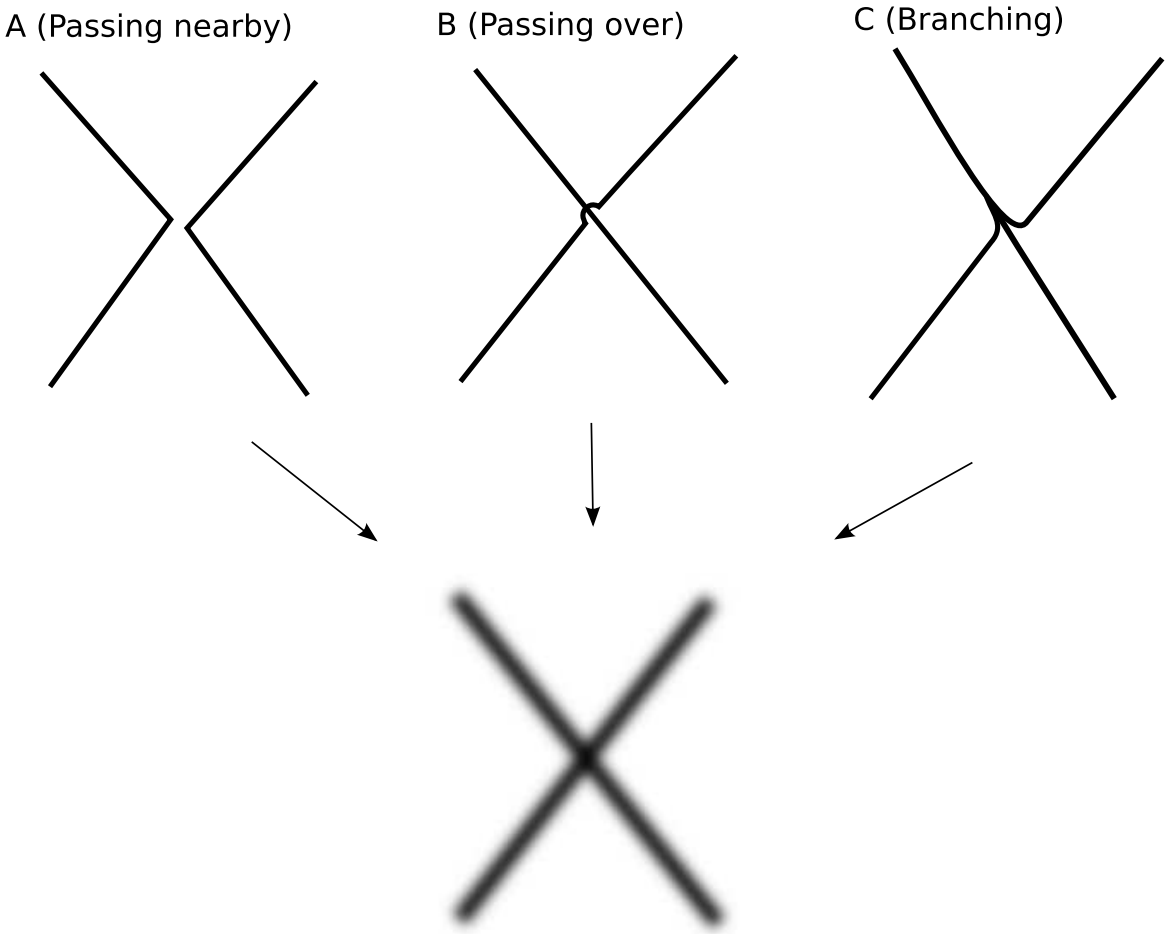


Figure 55 – A caricature of the problem of distinguishing branching neurons from those that cross over or just passes close by one another. All three of these distinct neuronal morphologies may end up being indistinguishable in the acquired image.

and perhaps in the future a *Drosophila* version of the "Brainbow" technique [Lichtman et al., 2008].[45] When such datasets are commonly available, the techniques developed here could in principle be applied for tracing neurons but with much less ambiguous results.

A further point to consider about our dataset is that it provides incomplete coverage even of the neurons which do fluoresce. This is partly because the cropping of the images around the central complex and mushroom bodies means that sections of the neurons may be missed,[46] and partly because the fluorescence may be very faint in some processes and so may create discontinuities where it drops too low. The former issue is an inevitable consequence of needing to acquire high resolution images with our current imaging protocol. In practice, however, this is not a serious problem as we are chiefly interested in the information flow between elements of the central complex and mushroom bodies, so if there are identifiable dendritic and axonal regions along a particular path, this will suffice. The latter issue can be largely addressed by replication, so we raise the probability that a faint neuron will appear completely in one of those scans.

There is another set of issues regarding tracing connectivity that was brought to my attention by users (and potential users) of the software I have developed: these are to do with the expectations that people have of the tools. The most important of these is that being able to trace out a path in an image is not in itself evidence of anything - it is possible, although quite tedious, to force these tools to pick out paths made largely of noise, which have no biological meaning. We can use the traces and various measures to estimate which are the most plausible connections, but these still need to be validated by human experts. Despite this and other limitations, having a computer-recorded specification of paths within a confocal stack has enormous advantages for describing neuroanatomy, including the following points:

- When publishing data describing a connection, a package of the image and trace files specifies precisely where the hypothesized connection lies, in contrast to a linguistic description in often ambiguous neuroanatomical terms.
- By aggregating traces from multiple brains we can look at the variability of the path with statistical measures.
- Certain types of summary statistics for the neurons are simple to calculate automatically,

[45] These alternatives are further discussed in section 7.1.1.

[46] This is particularly the case for some somata and processes leading to the optic lobes and thoracic ganglion.

such as the length and number of branches in a path. Others are possible after a further reconstruction phase, such as calculating the volume or surface area of reconstructed neurons.

- It is easy to generate images that clearly pick out paths of interest from noisy scans.
- With registration techniques and data from other markers we can annotate each path with measures of how likely a part of the path is to be in a dendritic region or an axonal region of the neuron. (See section 7.1.2.)
- Since we have multiple overlapping neurons in each of these scans, the correct interpretation and division of traces into separate neurons is often ambiguous. By recording all the morphology accurately, and largely without interpretation, we can reinterpret the data later.

In a way, the most important sense in which this type of computational neuroanatomy is an improvement on traditional 2D paper publication is the first of these points, i.e. that we can publish the complete source image data, with each detail annotated in 3D stacks.

As well as those advantages, which focus on the end product (i.e. the traced data), it is worth noting that acquiring the data from the expression patterns in these images is essentially impossible without computer assistance, due to the large number of neurons (and corresponding confusion) one has to deal with.

5.2 Background

The task of tracing neurons can be seen as trying to reduce a 3D image to a vector-based description of the structures shown. These vectors are chained into a graph structure, which might be restricted to a tree, a DAG (directed acyclic graph) or a full graph depending on the model used by the algorithm. As far as possible I try to separate this task from that of reconstructing the diameter and surface of neurons, only adjusting the paths and finding diameters at each point after the basic logical structure has been found.

There are several broad approaches to this problem, but I think the most helpful division is that suggested in [Al-Kofahi et al., 2008], between skeletonization-based techniques and "tracing" (or "exploratory") algorithms.

5.2.1 Skeletonization

The skeleton of a binary 3D image can be defined as the medial axes of the shape in question - it is made up of the centre points of the largest possible spheres which fit completely inside the shape while being tangential to the surface of the shape at at least two points, where "largest possible" means that there can be no larger sphere which completely contains it and still remains inside the shape [Lee et al., 1994]. The skeleton of a shape can be generated in a variety of ways, but many skeletonization or "thinning" techniques work by iteratively removing voxels uniformly from the edges of the structures in the image, being careful not to take away points that would change its topology, e.g. by separating one structure into two or more. Once the image consists only of the skeleton, it can be processed to locate the voxels which represent branches. At a high level, this is a pleasingly intuitive approach to reducing an image to traces, although the implementation details can be complex. An example of such an algorithm in 3D is described in [Lee et al., 1994], and this has been implemented by Ignacio Arganda-Carreras as the Skeletonize3D plugin for ImageJ. The problem typically cited with these approaches (e.g. in [Al-Kofahi et al., 2002]) is that they are computationally expensive, since they have to consider every point in the 3D stack. Two other considerations are that:

- With noisy data, many spurious structures are found even with careful preprocessing of the images, making curation necessary.
- Since such algorithms operate on binary images, the original data must be carefully thresholded to maintain the connectivity of the neurons without including too much noise or background; this may result in missing fainter branches.

Nonetheless, skeletonization may provide a useful approach to removing redundancy in my automatic tracing plugin. (See section 5.8.)

5.2.2 Tracing

The methods we describe as "tracing" algorithms (also referred to as exploratory algorithms) start at a particular point in the image and try to find an optimal path through the image from that point according to some suitable cost function and stopping criteria. Here I give a brief overview

of the scope of these methods, but for a more complete description of the pros and cons of such algorithms, the "Image Analysis Background" section of [Schmitt et al., 2004] provides a more useful and detailed discussion. To convey the scope of these techniques, however, some of the key respects in which they vary may include:

- What level of user interaction is required in order to trace a brain. Some methods have fully automatic modes, e.g. FilamentTracer (part of Imaris[47]), while with others it is necessary to specify start points, branch points, and so on.
- The extent to which the algorithm proceeds in 3D. For example, the method proposed in [Al-Kofahi et al., 2002] reduces the image to 2D for its initial detection of seed points, while other programs operate only on 2D images, e.g. NeuronJ [Meijering et al., 2004].
- Whether the values from the image data are used throughout the tracing and reconstruction process. For example, [Schmitt et al., 2004] points out that some methods (e.g. FilamentTracer) perform a final path smoothing technique that disregards the image data.
- How the path through the image is optimized. For example, NeuronJ [Meijering et al., 2004] uses a variation of Dijkstra's algorithm to find a path through the image, whereas other algorithms use variations of the "snakes" segmentation technique, [Kass et al., 1988] or the more complex active geodesic contours technique [Caselles et al., 1997].
- The model of neuron shape used. Typically this is a generalized cylinder model, but some algorithms use more complex models for detecting branch points, where the cylinder model is typically unhelpful, e.g. [Al-Kofahi et al., 2008].
- Filtering of the image to detect features of neurites. There are a variety of Hessian-based techniques for this, [Sato et al., 1998] others based on finding medial axes, [Pizer et al., 2003] or some simpler models for detecting the edges of neurons [Al-Kofahi et al., 2002].
- How (and if) branches are detected [Al-Kofahi et al., 2008].

Unfortunately, the most impressive reconstruction software available tends to be closed source. This includes the most widely-used ImageJ-based plugin, NeuronJ, a 2D neuron tracer developed by Erik Meijering et al [Meijering et al., 2004], which is free for academic use, but whose license does not

[47] `http://www.bitplane.com/`

allow researchers to examine or reuse the source code. NeuronJ allows one to pick a succession of points from a path in a 2D image with the additional help that while moving the mouse to decide on the next point the software shows its suggested best path between them. Thus, if the suggested path appears to be going wrong, one simply moves the mouse closer to the previous point along the intended path. NeuronJ uses a measure of "neuriteness" at each point of the image to define the best path through it, based on eigenvalues and eigenvectors of the Hessian matrix. (There is further discussion of such cost functions in the next section.) This mode of operation has the advantage that in noisy images an expert can still manually pick points that are closer together and still annotate paths.

The results from commercial or closed-source tools for fully- and semi-automatic neuron tracing, such as Neurolucida[48] and FilamentTracer, are very impressive, but the cost of the software is prohibitive. The "skeletonize" tool described in [Evers et al., 2005, Schmitt et al., 2004] [49] is free but no source code is available, and it depends on the expensive and reportedly unreliable Amira platform. Our justification for preferring to develop our own tool instead is the same as that set out in section 1.6: research into and development of these methods is inhibited when they are not released under a Free Software license.

5.3 Development (Simple Neurite Tracer)

5.3.1 Semi-Automated Tracing

The type of tool that I believed would be most useful for this project is a 3D analogue of NeuronJ, which I called "Simple Neurite Tracer". This was initially developed as an ImageJ plugin built from the VIB source code repository, but came to depend on so many other components within that repository that it became impractical to distribute a standalone version. The Simple Neurite Tracer is available bundled as part of Fiji, which enables developers to be sure that other dependencies (such as Benjamin Schmid's 3D viewer) will be present. I am indebted to the other core developers for their work on the Fiji project, and in particular those whose code contributes to this plugin, such as Dr Stephan Preibisch, Dr Albert Cardona and Dr Johannes Schindelin. (Further acknowledgments and details can be found via `http://fruitfly.inf.ed.ac.uk/~mark/phd/` and in Chapter 8.)

[48] `http://www.mbfbioscience.com/neurolucida/`

[49] The binary module for Amira can be downloaded from: `http://www.neurobiologie.fu-berlin.de/pflueger/evers.html`

5.3.2 Search Implementation

Searching for paths through an image stack can be seen as the same problem as the classic computer science question of how to find the shortest path between two nodes in a graph. To make this identification, we consider the grid of sample points in the image to be nodes in the graph and assume that there are connections in both directions between any two adjacent points. In this section I first consider the question of appropriate cost functions to associate with moving from one point to another and subsequently discuss the precise search method implemented.

Cost Functions: I tested several different cost functions[50] in the course of developing the Simple Neurite Tracer plugin. In every case, however, the cost of moving from point a to point b in an image is based on a value calculated at point b, which is multiplied by the Euclidean distance between a and b. This means that the sometimes different separations between a point and the 26 around it are taken into account. The different options considered are:

- Linear scaling of the intensity at point b. In the case of 8 bit images, this is just $255 - b$.
- Reciprocal scaling of the intensity at point b. If the intensity value at point b is $I(b) \in [0, 255]$ then the cost C in moving to b is:

$$C(b) = \begin{cases} 2 & \text{if } I(b) = 0 \\ 1/I(b) & \text{if } I(b) \neq 0 \end{cases}$$

- Reciprocal scaling of a Hessian-based "tubeness" value. This is explained in the next section.

The first of these options (simple linear scaling) is substantially less effective than the other options, so is not presented in the user interface of the tracer, although it is available through trivial changes to the source code.

Hessian-based "Tubeness" [Meijering et al., 2004] uses the word "neuriteness" to describe the measure that NeuronJ searches on within the image. This measure is described as reflecting the

[50] It is tempting to use the word "metric" to describe these cost functions since they seem analogous to distances, but this is misleading: we typically use functions that break both the symmetry and transitivity requirements that a metric must meet.

likelihood that a point in the image is within a tube-like structure. Here the term "tubeness" is used to describe the same concept, although elsewhere in the literature these terms and "vesselness" are used synonymously. These measures are often some function of the 3D curvatures of the image at a particular point, based on the insight that at a point in the image which is tube-like in the surrounding area, the axes of greatest variation will extend perpendicular to the major axis of the tube, while parallel to the major axis of the tube there will be very little variation. Measures of variation in these axes which are often used are the eigenvalues of the Hessian matrix, where the eigenvectors give the direction of these axes. The Hessian matrix H for a continuous intensity image function $F : \mathbb{R}^3 \rightarrow [0, 255]$ is defined as:

$$H(F) = \begin{pmatrix} \frac{\partial^2 F}{\partial x^2} & \frac{\partial^2 F}{\partial x \partial y} & \frac{\partial^2 F}{\partial x \partial z} \\ \frac{\partial^2 F}{\partial y \partial x} & \frac{\partial^2 F}{\partial y^2} & \frac{\partial^2 F}{\partial y \partial z} \\ \frac{\partial^2 F}{\partial z \partial x} & \frac{\partial^2 F}{\partial z \partial y} & \frac{\partial^2 F}{\partial z^2} \end{pmatrix}$$

If the eigenvalues of this matrix at a particular point are $\lambda_1, \lambda_2, \lambda_3$, ordered $\lambda_1 > \lambda_2 > \lambda_3$, as explained in [Sato et al., 1998] we would expect λ_1 to be close to zero and λ_2 and λ_3 to be large and negative at tube-like points in the image. (This is assuming that points within the neuron have higher values than those outside - otherwise the eigenvalues are positive rather than negative.)[51] [Sato et al., 1998] provides a useful review of several ways of using these eigenvalues to pick out such features, but for the sake of simplicity we have chosen to use $\sqrt{\lambda_2 \lambda_3}$. To produce the cost function from this, as mentioned above, we take the reciprocal, so in fact use the measure:

$$\frac{m}{\sqrt{\lambda_2 \lambda_3}}$$

... where m is a constant multiplier chosen to scale values from the function to a similar range as the reciprocal intensity measure. (This is necessary because of the inverse scaling: otherwise a constant multiplier would make no difference.)

Another consideration with this measure is that we convolve the image with a 3D Gaussian function in order to smooth the image and determine the scale of features that we wish to pick out, based on the idea of scale-space representations of images [Lindeberg, 1994]. The standard deviation of the Gaussian kernel used, σ, is another parameter that the user should choose carefully. (A simple

[51] I am using code written by Stephan Preibisch to calculate the Hessian matrix and the curvatures.

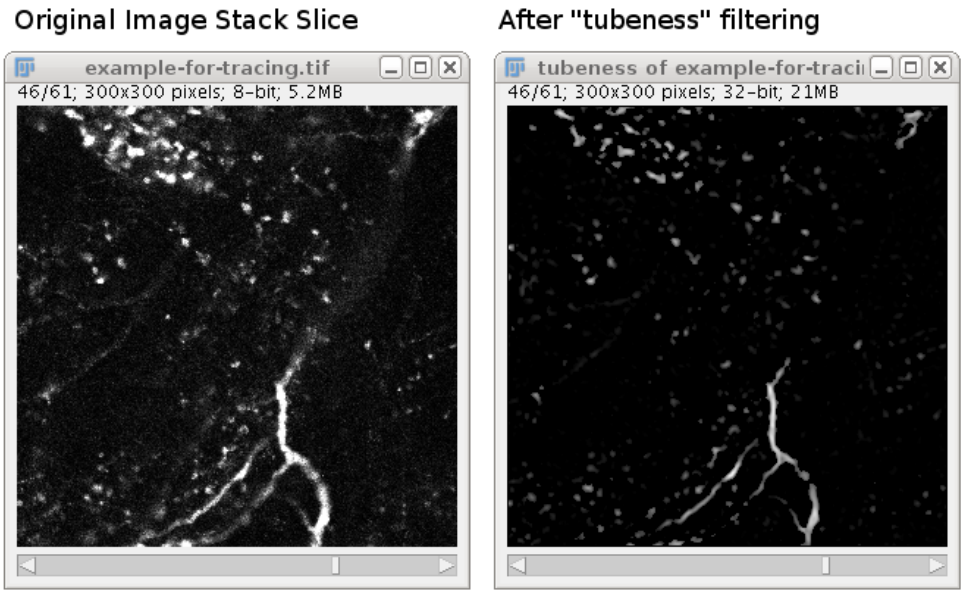


Figure 56 – An example of "tubeness" filtering based on eigenvalues of the Hessian matrix. (Note that the processed view of the image is not shown in Simple Neurite Tracer - instead it is calculated largely on the fly as needed while searching out paths. To generate such filtered versions of image stacks, one can use the Tubeness plugin, which is based on the same source code.)

interface for doing this is shown in the "Preprocessing Options" section of section 5.3.4.) An example of the effect of this tubeness filtering on an image stack is shown in Figure 56.

Search Methods: One of the standard algorithms for dealing with this search problem with a single start point is known as Dijkstra's algorithm [Cormen et al., 2001], which finds the shortest path from a single source point to all other points in the graph. However, there are many variants of this type of "best-first" search algorithm appropriate under different conditions, and for this tracing problem we have the additional help of a defined goal. The implication of this is that we can use some heuristic estimate of the distance to the goal to inform the algorithm about which search paths are most promising. The A* search algorithm [Hart et al., 1968] is a classic heuristic search algorithm which uses such an estimate: the points to explore next are chosen based on a value $f(x)$ at each point x: this is the sum of $g(x)$, the shortest distance found to x so far, and $h(x)$, an estimate of the distance from x to the goal. This is a standard algorithm to use for such heuristic search problems, but it also has two particular advantages which may not be obvious:

- If $h(x)$ is set to 0 for all points, the algorithm reduces to Dijkstra's algorithm. This means that the same code can be reused in order to deal with the case where there is no particular defined goal point, which we use later in automated tracing techniques.
- There exists a useful study on the performance of variants of the A* algorithm in path-finding that used biological image data as a test set [Wink et al., 2000].

The results in [Wink et al., 2000] suggest that performing a bi-directional version of A* search may improve performance in the 3D case where reciprocal intensities are used as the distance measure. In many cases the number of nodes that the algorithm needs to explore will be much lower if we start searches from both the start and the goal, terminating if we explore a node in one of these searches that is already present in the closed set of the other. At each step of the A* algorithm we choose to extend whichever search has the fewer open nodes, i.e. those on the "frontier" of the search, as suggested in the Wikipedia article on bi-directional search.[52] The heuristic that I have used is simply the Euclidean distance from the point to the goal scaled by the worst case cost; this ensures that the heuristic is admissible, i.e. that it cannot underestimate the minimum cost required to reach the goal from that point.

5.3.3 Reconstructing Neuronal Volumes

In this description so far, we have been modelling neurons as one-dimensional structures: in other words, we consider them to be paths with zero width and volume. In some respects this is all that we may need, but several considerations suggest that some facility for reconstructing the volumes of neurons from these paths is useful:

- For visualization and producing publication quality images, it is much more realistic to generate volumes as well as paths.
- In fitting a volume around the path we can optimize the path to pass through the midpoint of the neuron. This should make the estimates of lengths of neurons more accurate as well.
- Statistics about the volume of these structures may be of interest.

[52] `http://en.wikipedia.org/wiki/Bidirectional_search` on 2009-06-29.

As suggested above, there are many methods that have been employed for this type of reconstruction. Of particular note are level sets methods, which work by allowing a fluid interface around the path to evolve to find the boundary [Sethian, 1999] or model-based reconstructions which look for the edges of neurons with an appropriate kernel [Al-Kofahi et al., 2002]. We approached this reconstruction task with two different methods, described in the sections "Simple Neuron Filling" and "Fitting Radii and Midpoints" below.

Simple Neuron Filling: This useful and easy-to-implement method is similar to the fast-marching technique described in [Sethian, 1999], but with boundaries limited to voxel-level accuracy. Essentially, this method uses Dijkstra's algorithm to explore how costly it is to move from any point on the path to another nearby point, and then allow the user to pick a maximum threshold value which defines the points considered to be in the neuron. The cost function used, however, takes into account the intensity values so that this threshold will define a volume that fits the neuron better than a fixed radius tube, the structure which would be defined by a pure Euclidean metric using this method. The cost function, consistent with the basic tracing metric described above, is:

$$f(\mathbf{x}, \mathbf{y}) \rightarrow \frac{|\mathbf{x} - \mathbf{y}|}{I(\mathbf{y})}$$

...where $I(y)$ is the image value (scaled to be in $[0, 255]$) at point y. By picking the threshold carefully with this method one can sometimes achieve good reconstructions. Another pragmatic use of the fills created with this method is that they can be used as a quick way to extract a single structure from a noisy image, and in many of these use cases (visualization, for example) setting the threshold too high may not matter.

However, despite its pragmatic advantages, this method has the following problems:

1. It does not produce accurate enough reconstructions of neurons to make it useful for volumetric analysis - in particular it obviously gives poor results whenever there is significant variation in the radius of the neuron while the intensity of values within the neuron remains similar.

2. Currently a manual step is required to pick a threshold value. While this can be done with a simple mouse-click, we would prefer this to be automatic.

3. It does not immediately help with optimizing the path to match midpoints of the neurons.

Fitting Radii and Midpoints: Motivated by the deficiencies of the previous method, I also implemented a model-based approach to reconstruction. This attempts to fit a circular cross-section to each point in the path by optimizing the radius and the centre point in a plane normal to the tangent of the path. A more detailed description follows below.

Defining the Tangent Vectors: If we have a path of n points where the positions of the points are $\mathbf{x}_\mathrm{i} : i \in [0, n)$, at each point we define the tangent vector $\mathbf{t}_\mathrm{i}$ to be:

$$\mathbf{t}_\mathrm{i} = x_{\min(i+w,n-1)} - x_{\max(i-w,0)}$$

... where w controls how far away the points to base the tangent on should be - we typically set this to 4. This reduces the effect of small local variations causing misleading tangents to be used to define the normal plane.

Generating the Normal Plane: The tangent vector defines a normal plane, but we only consider values in a small square around the original path point. Taking into account the voxel separation and using trilinear interpolation we produce an $s \times s$ square of values, which correspond to points in that normal plane separated by the minimum voxel separation. (The rotation of this square section within the plane is essentially arbitrary, since we set s to 40, which easily contains any neurons found in this set of images.)

Fit a Circle: The parameters we now try to optimize are $[x, y, r]$ where x and y are the centre of the neuron's cross-section in this plane and r is the radius of the neuron. Of course, some of the neurons we find in these images are far from circular in cross section, but this simple model works well in most cases. The two candidate functions I tried to optimize were the (a) Circular Difference Fitting Function, and (b) Mexican Hat Convolution Fitting Function.

Circular Difference Fitting Function: If v_{max} is the maximum value within this normal plane and $N(x, y) \in [0, v_{max}]$ is the intensity value of the point at x, y in the normal plane, then we define a function $c_p(x, y, r, x_i, y_i)$ as the following:

$$c_p(x, y, r, x_i, y_i) := \begin{cases} N(x_i, y_i) & \text{if } (x - x_i)^2 + (y - y_i)^2 > r^2 \\ v_{max} - N(x_i, y_i) & \text{if } (x - x_i)^2 + (y - y_i)^2 \leq r^2 \end{cases}$$

Then the circular difference function $c(x, y, z)$ is defined as:

$$c(x, y, z) := \sum_{(x_i, y_i) \in \{points\, in\, normalplane\}} c_p(x, y, r, x_i, y_i)$$

In other words, it is the sum of all values outside the circle and v_{max} minus each of the values inside the circle. This will clearly be minimal in the perfect case, but there are some obvious objections. In particular:

- There tends to be some fuzziness towards the edge of the neurons.
- If we find a neuron-like cross-section at a point in the plane, it is not obvious that the values in the rest of the plane should affect the score - they may be completely separate anatomical structures.

A simple alternative to this model (suggested to me by Dr Ting Zhao) is to fit a so-called "Mexican Hat" function, as described in the next section.

Mexican Hat Convolution Fitting Function: The "Mexican Hat" function is the negated second derivative of a Gaussian, defined as:

$$\psi(t) = \frac{1}{\sqrt{2\pi\sigma^3}} \left(1 - \frac{t^2}{\sigma^2}\right) e^{\frac{-t^2}{2\sigma^2}}$$

We use this with $t = \sqrt{(x - x_i)^2 + (y - y_i)^2}$ and $\sigma = r$ to give the alternative function:

$$m_p(x, y, r, x_i, y_i) := \frac{-N(x,y)}{\sqrt{2\pi} r^3} \left(1 - \frac{(x-x_i)^2 + (y-y_i)^2}{r^2}\right) e^{\frac{-((x-x_i)^2 + (y-y_i)^2)}{2r^2}}$$

And then, as before, summed this value at each point in the normal plane to give:

$$m(x, y, z) := \sum_{(x_i, y_i) \in \{points\, in\, normalplane\}} c_p(x, y, r, x_i, y_i)$$

In practice, however, this cost function produced poor fits of cross-sections in the normal plane, having a tendency for the minimized solution to include too large a region of the image, particularly if there were several blob-like features close together.

Optimization Method: The optimization method used to minimize these cost functions was the Conjugate Direction Search from the `pal` library[53], based on a optimization method suggested by Prof Richard P. Brent. The start values used at each point were the x and y at the centre of the normal plane (corresponding to the path's point found in the original search) and an initial value of r which is set to 3 (corresponding to 3 times the minimum voxel separation).

Mark Outliers as Invalid: Typically, a number of these optimizations will fail to find the correct neuronal cross-section in the normal plane. We attempt to detect and discard these outliers by taking as an estimate of the "true" neuron's radius at each point the median optimized radius within the nearby region of the neuron (specifically looking at up to 4 points on either side), and if the centre point has moved further than this distance in the course of optimization we mark that point as invalid.

The next filtering applied to the optimized points is to discard any points where the angle between that point and the two on either side is more acute than 90°. This removes a number of points where a mis-fitted radius causes an unexpected kink in the neuron, but has the problem (as mentioned in [Schmitt et al., 2004]) that it will smooth out genuine sharp changes of direction.

Finally, we repeatedly go through the list of fitted circles and for each one count how many of the other circles it intersects with. Then we iteratively mark as invalid the circles that overlap most with other ones. (If two with the same "overlap count" are adjacent we remove the one with the larger radius on the basis that large misfits tend to be worse than smaller ones.) This process is repeated until there are no overlapping circles left marked as valid.

Filling in Gaps: The previous step may leave large gaps where the fitting is very poor, so if more than a particular number of points in a row are marked as invalid, we replace the results of the optimization at some of these points with a circle of radius equal to the minimum voxel separation. (Cases where the optimization fails at many points in a row are typically very faint processes in the image, so this turns out to be a reasonable way of filling in these values.)

Path Smoothing: No further path smoothing is applied to the data in the tracer, although when added to the 3D tracer (using the `makeTube` function from TrakEM2) some smoothing is applied to

[53] `http://pacific.mpi-cbg.de/cgi-bin/gitweb.cgi?p=VIB.git;a=blob;f=pal/math/ConjugateDirectionSearch.java`

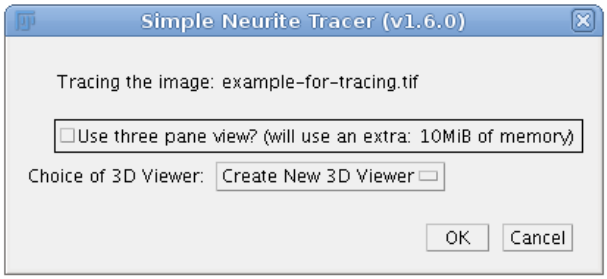


Figure 57 – The initial dialog for Simple Neurite Tracer. The second option only appears on systems that support Java 3D.

improve the rendering.

5.3.4 Tracer Interface Considerations

One of the challenges in developing a tool such as this tracer, which supports a number of different workflows and many options, is to make the user interface easy to navigate. In this section I briefly describe the most important features of the user interface.

Initial Dialog: The first dialog that a user is presented with on running the plugin is shown in Figure 57. If the user selects the three pane view, then later in the tracer they will be presented with the image stack sliced in not only the standard XY plane, but also YZ and XZ, as shown in Figure 58. The 3D viewer option, which is recommended by default, means that the tracer will reflect its progress in Benjamin Schmid's Java3D-based viewer for ImageJ. If an existing 3D viewer is open then it is possible to reuse the viewer instead of creating a new instance, which is very valuable for building up complex 3D scenes. For example, on a single run of the tracer one can add paths in a particular colour, and then on a subsequent run add further paths distinguished by a different colour.

Main Dialog: The main dialog for Simple Neurite Tracer is shown in Figure 60. The most important feature of this interface is the "Instructions" box in the top left hand corner of the screen. This should always provide guidance on what the next expected step in the process of tracing an image is. The buttons underneath it typically offer choices associated with those instructions.

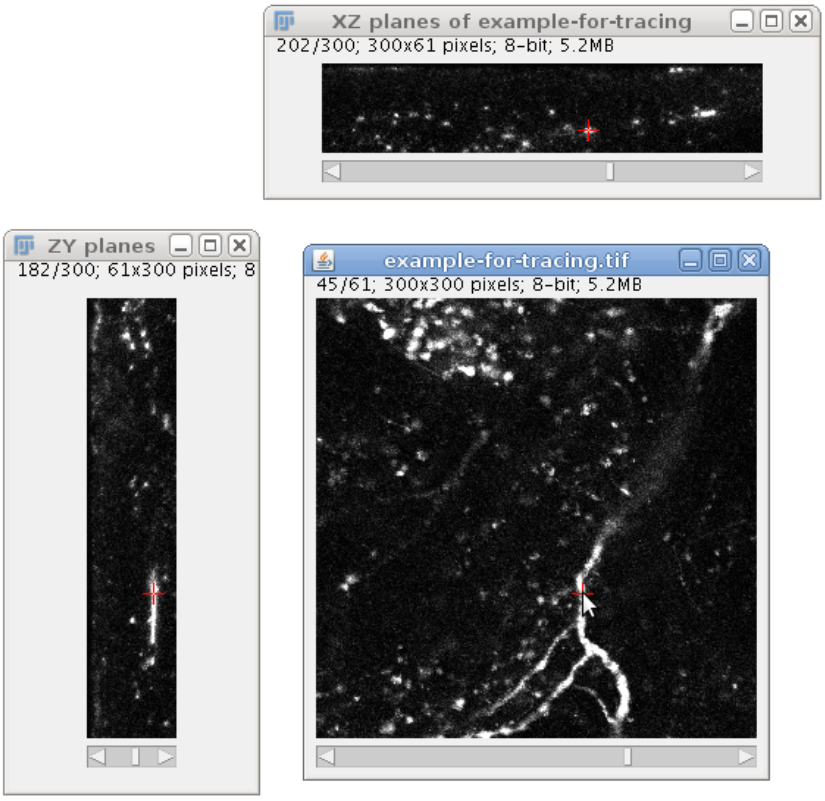


Figure 58 – The three pane view offered by Simple Neurite Tracer.

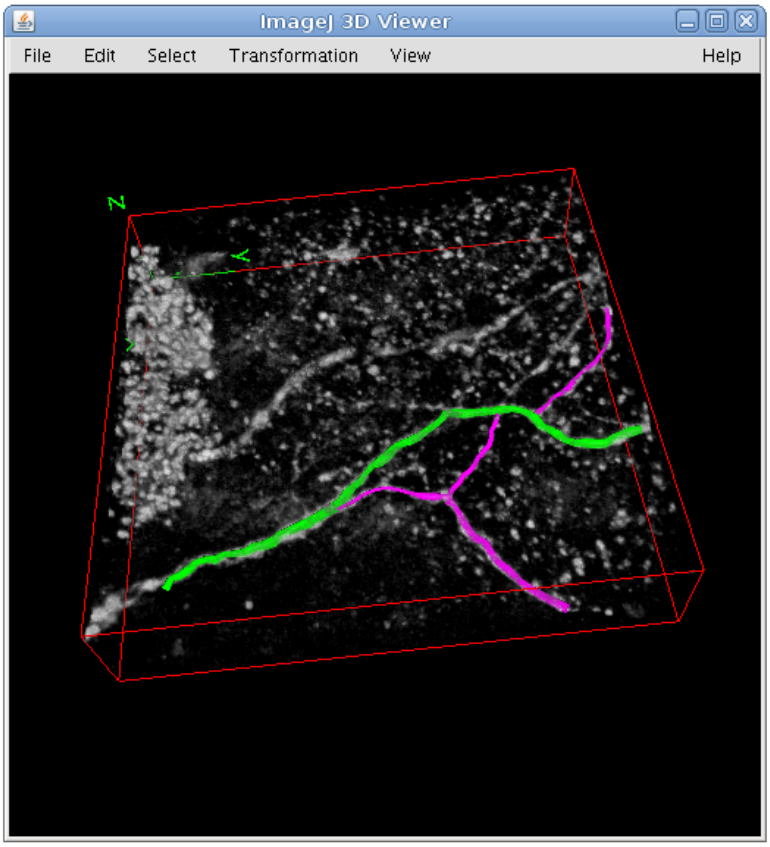


Figure 59 – Benjamin Schmid's 3D viewer, used to display results of the tracing.

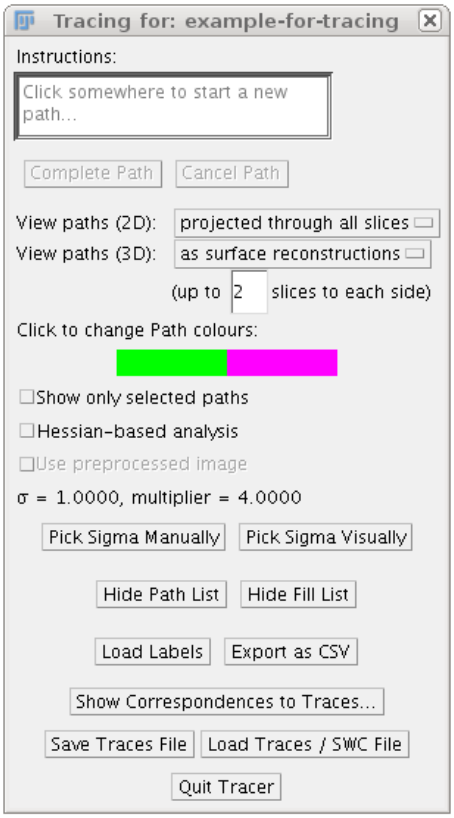


Figure 60 – The main dialog of Simple Neurite Tracer.

Below those elements of the user interface are two options that relate to the presentation of the tracer in the 2D and 3D views. In the first case, one is offered the option of seeing paths projected through all slices or just the elements of paths that are close to the current slice. The difference between these options is shown in Figure 61. In the second case (3D view options) one is offered the choice of viewing paths in one of the forms shown in Figure 62.

The user can click on the two coloured boxes below to change the colour of all selected and deselected paths. The "Show only selected paths" option allows the user to temporarily remove uninteresting paths from both the 2D and 3D view.

Preprocessing Options: The "Hessian Based Analysis" option allows the user to turn on the Hessian-based "tubeness" filtering explained in section 5.3.2. It will use the parameters for σ and m ("multiplier") shown in the dialog two lines below. Those values can be changed manually by clicking on the "Pick Sigma Manually" button, or the more user-friendly "Pick Sigma Visually" interface option. This latter button allows one to click on an interesting point in the image to produce a small preview palette with the region around that point preprocessed with values of σ from half the minimum voxel separation to 4.5 times the minimum voxel separation. This is shown in Figure 63. The "Use preprocessed image" option is available if, when starting the plugin for an image file `[basename].tif`, a file `[basename].tubes.tif` is found to exist. If the "Use preprocessed image" option is then selected, the values from the `[basename].tubes.tif` image will be used as the basis for the cost function - this allows an advanced user the option to:

- Save time by preprocessing images in a batch before tracing them. (The process of calculating the Gaussian convolution of the image can take a long time on slower computers so this may be worthwhile.)
- Experiment with different "tubeness" filters.

The two buttons "Hide Path List" and "Hide Fill List" are for toggling the visibility of the Path List and Fill List windows. The latter provides an interface for controlling the fill-based reconstructions of paths described in section 5.3.3; this window is shown in Figure 65. The Path List window (Figure 64) is more important for most users: this window displays the tracing progress so far with the logical relationships between the paths. Since every path can begin or end on any other path, the connectivity representable in the tracer is a full graph, but for the purposes of displaying this in a

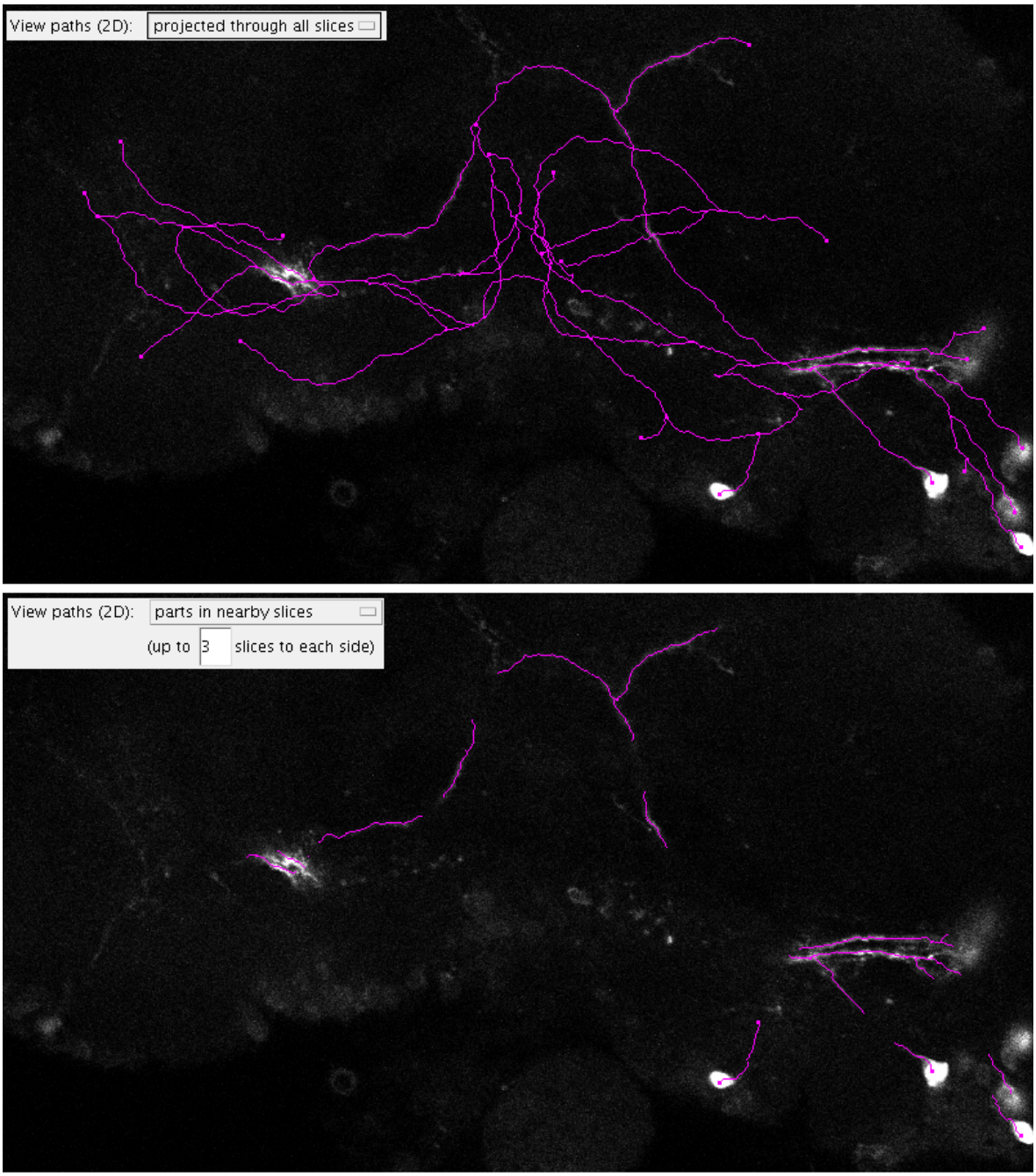


Figure 61 – This figure shows the difference between the "projected through all slices option" (top) and the same view with displayed path elements limited to those in slices up to 3 on either side of the current slice.

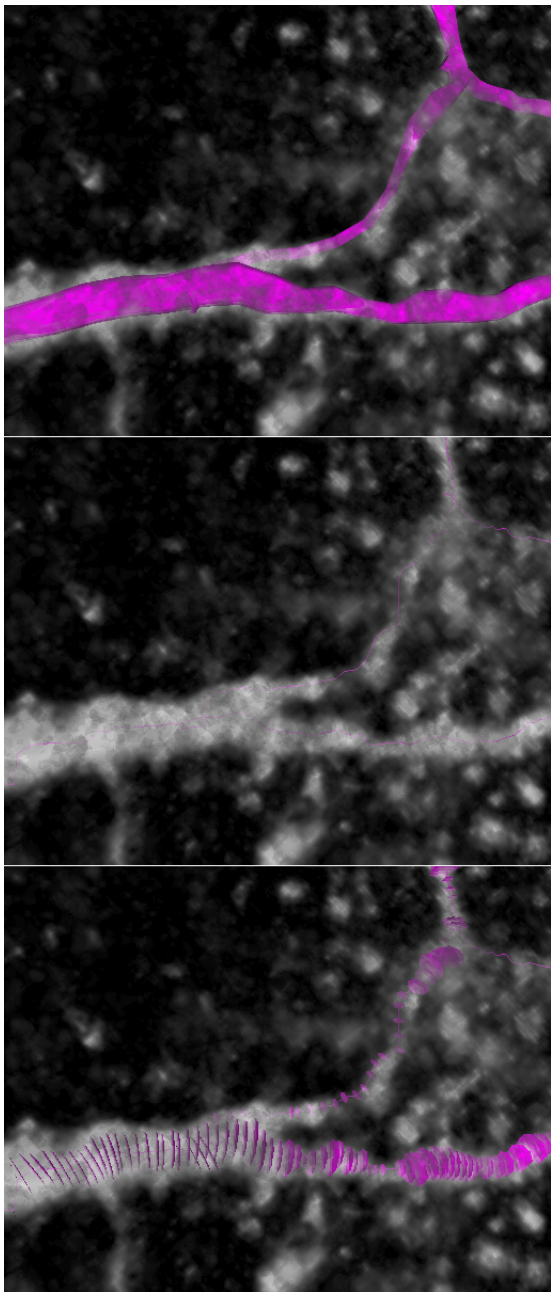


Figure 62 – This figure shows the difference between the three different 3D view options. The top view shows the "as surface reconstructions" option, the middle view shows the "as lines" option (the lines are a single pixel wide, so difficult to make out in this image) and the bottom view shows the "as lines and discs option". In the lattermost case, the discs will only be shown for paths that have had "Fit Volume" applied to them; otherwise they will appear as in the middle view.

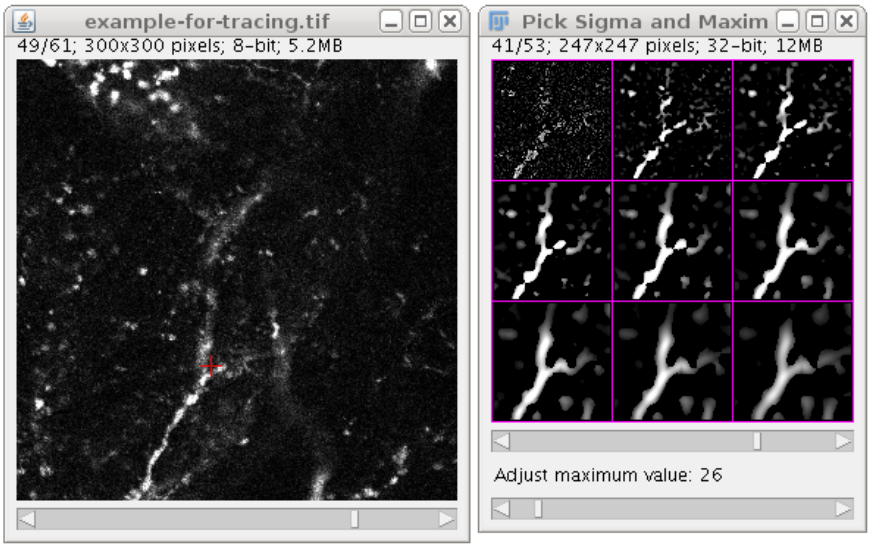


Figure 63 – The sigma palette or "Pick Sigma Visually" option. The lower of the two sliders allows the user to adjust the multiplier value. The upper of the two sliders selects the level in the preview stack. An appropriate value of sigma can be selected by clicking on the appropriate pane of the preview and then closing the window. (This is prompted in the "Instructions" box as usual.)

clear way in the Path List, each separate connected set of paths is shown as a tree, initially rooted at the first created path in that set. If the user realizes that it makes more sense for a different path to be considered the primary path in that set, they can select the path and click "Make Primary", which will cause those connected paths to be "re-rooted" with the selected path at the top level. The option "Fit Volume" or "Un-fit Volume" toggles between the fitted and un-fitted versions of the path, using the fitting algorithm described in section 5.3.3. The "Fill Out" button launches the filling process (section 5.3.3) which can be controlled from the Fill List window.

5.4 Validation (Simple Neurite Tracer)

A tricky issue with developing tools such as the Simple Neurite Tracer is how to validate the results that are produced. Ideally one would want to compare the results of semi-automatic tracing with the "real" topology of the neuron, but there is no such ground truth to compare our data to. (For example, even the concept of the centre line of a neuron is an artificial construct.) However, we would like some way to demonstrate that the results from this tool are reasonable. A few of the possible options are given below:

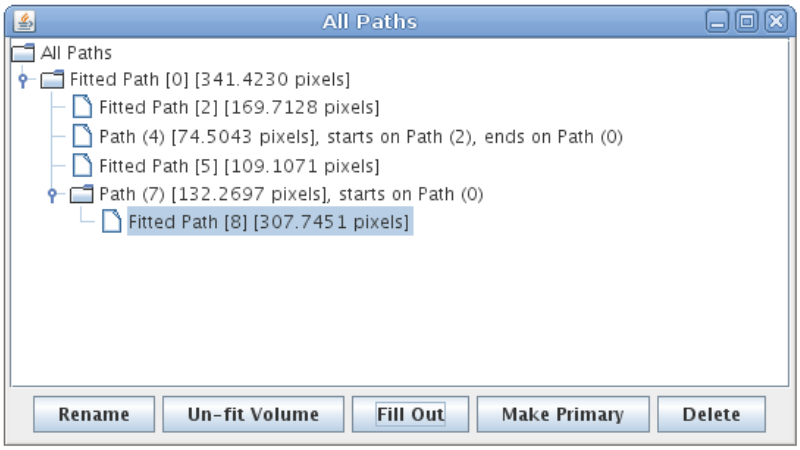


Figure 64 – The Path List window in Simple Neurite Tracer

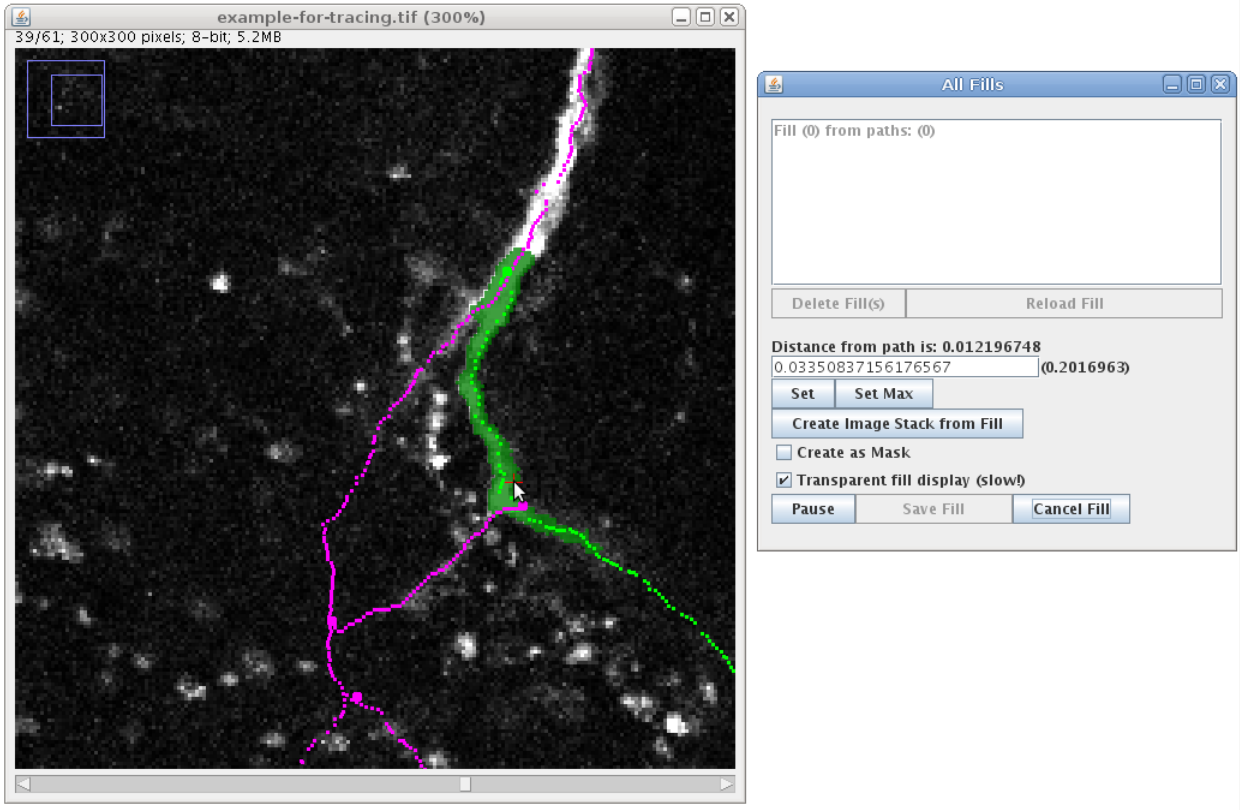


Figure 65 – The fill window of Simple Neurite Tracer

1. Generating a test set of synthetic images of neurons by some algorithm based on a defined centre line. Of course, this will only be as convincing as the algorithm to generate the neurons, and that would need some validation in the first place.

2. Repeatedly imaging the same structure from several angles (or just transforming one acquired image), and checking that similar results are produced no matter which of the images is used. This is an elegant idea used in [Al-Kofahi et al., 2002], but one which only checks for self-consistency of the algorithm.

3. Comparing the results to tracings produced by another tool. Of course, one cannot tell from this which results are a more accurate representation of the neuron's topology.

Dr Gregory Jefferis suggested that the data from `http://flybrain.stanford.edu` [Jefferis et al., 2007] would be useful for the last of these approaches. The tracings of these neurons were generated either manually in Neurolucida or hxskeletonize, the Amira module described in [Schmitt et al., 2004]. As a test set I traced 18 neurons from the "1" and "d" sections of this data with my own tracer, producing fitted and non-fitted versions of the traces. I then compared these 36 traces to the original 18, which were downloaded in SWC format.

The method employed to compare two traces for a particular image was to work out which points have corresponding points in the other traces file, and calculate statistics based on the differences between all of these pairs. Defining a "corresponding point" could be done several ways, but the method I used is as follows, assuming we have an arbitrary point in space a and want to find a corresponding point in the set of paths B, where each path is an list of points connected by lines:

1. Find the Euclidean distance between the point a and every defined point on the paths in B.

2. Pick the closest point to a, called b, say. If the closest point is further away than some defined limit, stop - there is no corresponding point.

3. The point b may have up to two line segments on either side, defined by the adjacent points on that path. For each of these, drop a perpendicular from a onto the line that contains that segment. If this meeting point is within the line segment on the path, it may still constitute a corresponding point; otherwise discard that line segment from consideration. If there is more than one such point found in this way, pick the one which is closer as the corresponding point

and stop. If there is only one such point found, that is the corresponding point. If no such points are found, continue to the next step.

4. If the point b has two adjacent points (i.e. the path has line segments on both sides of that point) then step 3 may fail to find a point where a is near to b and the path curves away from a at that point. So, if b has two adjacent points and a lies between the two planes each defined by (a) a normal vector from b to the adjacent point and (b) that adjacent point lying in the plane, then we consider b itself to be the corresponding point and stop.

5. If no corresponding point was found in the steps 3 or 4, b is discarded, and we return to step 2 to try the next closest point.

Examples of corresponding points found via this algorithm for a simple three-node path are shown in Figure 66. An intuitive way of looking at this idea of correspondence is that a correspondence is found either in a cylinder expanded around a segment of a path or in the "crack" between two such cylinders which are angled away.

It is important to note that this way of finding correspondences is asymmetric, i.e. the input is a point and a set of paths, returning a point lying somewhere on that set of paths. The implication of this is that the correspondences between every defined point in traces file A and traces file B are going to be very different from the correspondences between every defined point in traces file B and traces file A. In this validation section, the SWC files have relatively sparse points compared to the traces files created by my software, so we always use as input a particular point in the SWC file and the complete set of my traced paths. A much-enlarged close-up of a section of traces with the correspondences found in both directions can be seen in Figure 67.

Figure 68 shows for each traces file the standard deviation of offsets of corresponding points, comparing points from each SWC file with points on both the fitted and un-fitted paths I traced. The distance limit for corresponding points was set to 2.5μm. As a context for these numbers, the diameter of the neurons in these images files is typically about 1.5μm in the XY plane in the primary neural process, and rather larger in the branches that extend from them. In the Z plane the extent of the neurons can be as much as 5μm. So, compared to the imaged size of the neurons in question, these deviations are relatively small. Furthermore, it is not clear in many of the examples that I looked at which trace would be preferred; an example of this can be seen in Figure 67. While we cannot make very strong statements about the validity of the traces found using Simple Neurite Tracer based on these

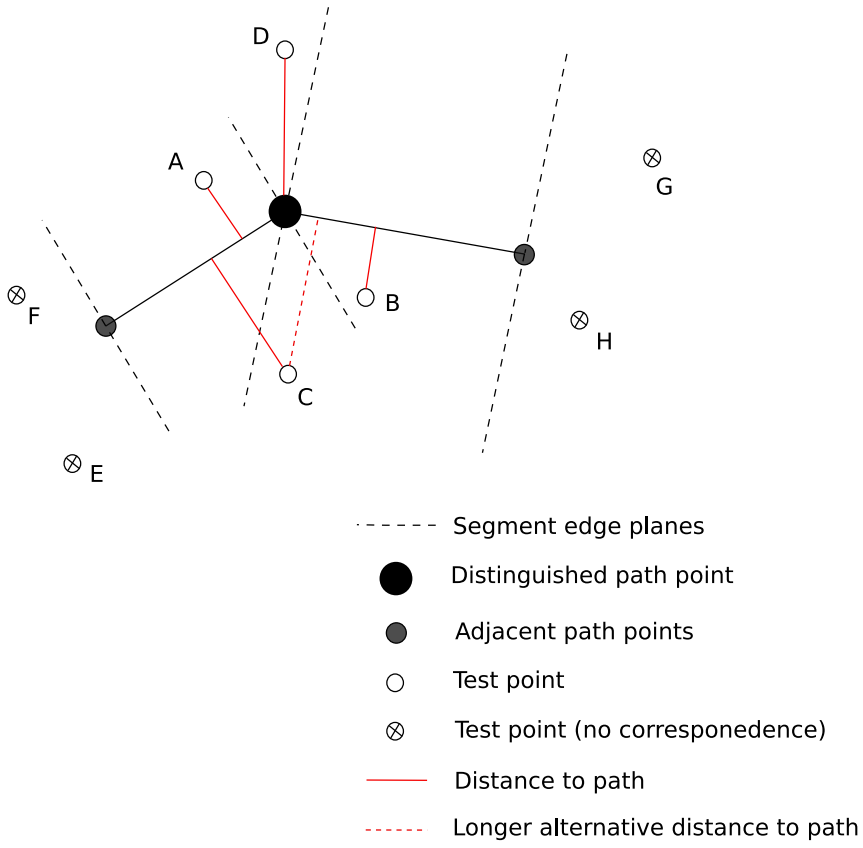


Figure 66 – Some diagrammatic examples of the algorithm for finding corresponding points described in section 5.4, reduced to 2D for simplicity of representation. Points A, B, C and D are all considered to have corresponding points on the path shown by solid black connected nodes. The corresponding point in each case is the other end of the solid red line. Point D is included because while perpendiculars dropped onto either line are outside the line segment, it lies between the planes defined by the points at the distal end of each line segment and the normal vector along that line segment. (That is case 3 in the algorithm.) Points E, F, G and H are considered to have no corresponding point on the path.

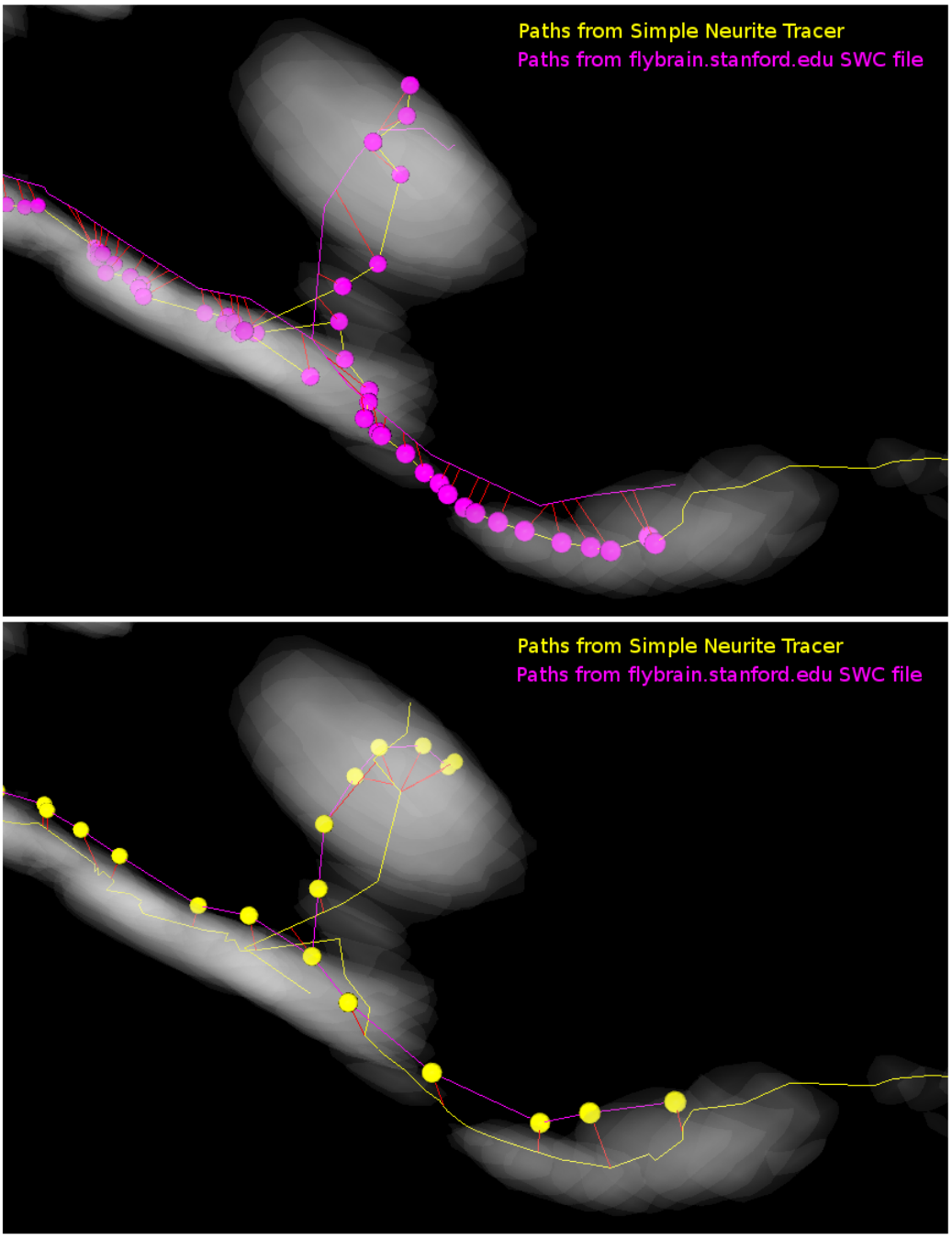


Figure 67 – Two images demonstrating the asymmetry of calculated "corresponding points" between traces files. The yellow paths were found with Simple Neurite Tracer (the tool described here) while the magenta paths are from an SWC file from `flybrain.stanford.edu`. The coloured balls show the point from which correspondences are found when starting from each of these. The red line shows the nearest line to what is considered a corresponding point. There are no corresponding points towards the right of the visible field, since the trace in the SWC file does not trace out the complete neuron. To give an indication of scale, the distance between the extreme left and right yellow balls is approximately $20\mu m$. (Since this is a projection into 2D it may not be clear that some of these correspondences are correct, but in the 3D viewer this can be checked by rotating the viewpoint.)

data, they provide some reassurance that the results are likely to be comparable to those produced by existing tools. Figure 69 shows what percentage of points in the SWC had correspondences that could be found in the fitted and unfitted traces produced with Simple Neurite Tracer. The small proportions of missed correspondences typically arose from tracing small subsidiary processes differently, or missing them completely in my annotation.

5.5 Results (Simple Neurite Tracer)

This section discusses the results of using the Simple Neurite Tracer tool to annotate the image corpus described in Chapter 2.

The process of tracing paths in each image takes between 30 minutes and 2 hours depending on the complexity of the scan. It is important to acknowledge that such manual tracing is not going to be *complete* in the sense that it is up to the annotator to carry on tracing paths until they are happy that they have covered the important connective features in the image stack. The tables below show summary statistics of the number of paths, the length of paths and branch points which I marked in each image. Those lines which are in italics represent brains where elements of the central complex appeared distorted even after the elastic registration was applied. We do not exclude them solely on this basis, however, since in many cases the neuron type which we look at later does not pass through one of these distorted regions. Misregistrations that affect particular neurons are very clear from the visualizations and graphs of variation shown later. The large variation in these summary statistics is to be expected: due to differences in the qualities of the scans, there are many more processes identifiable in some scans than others.

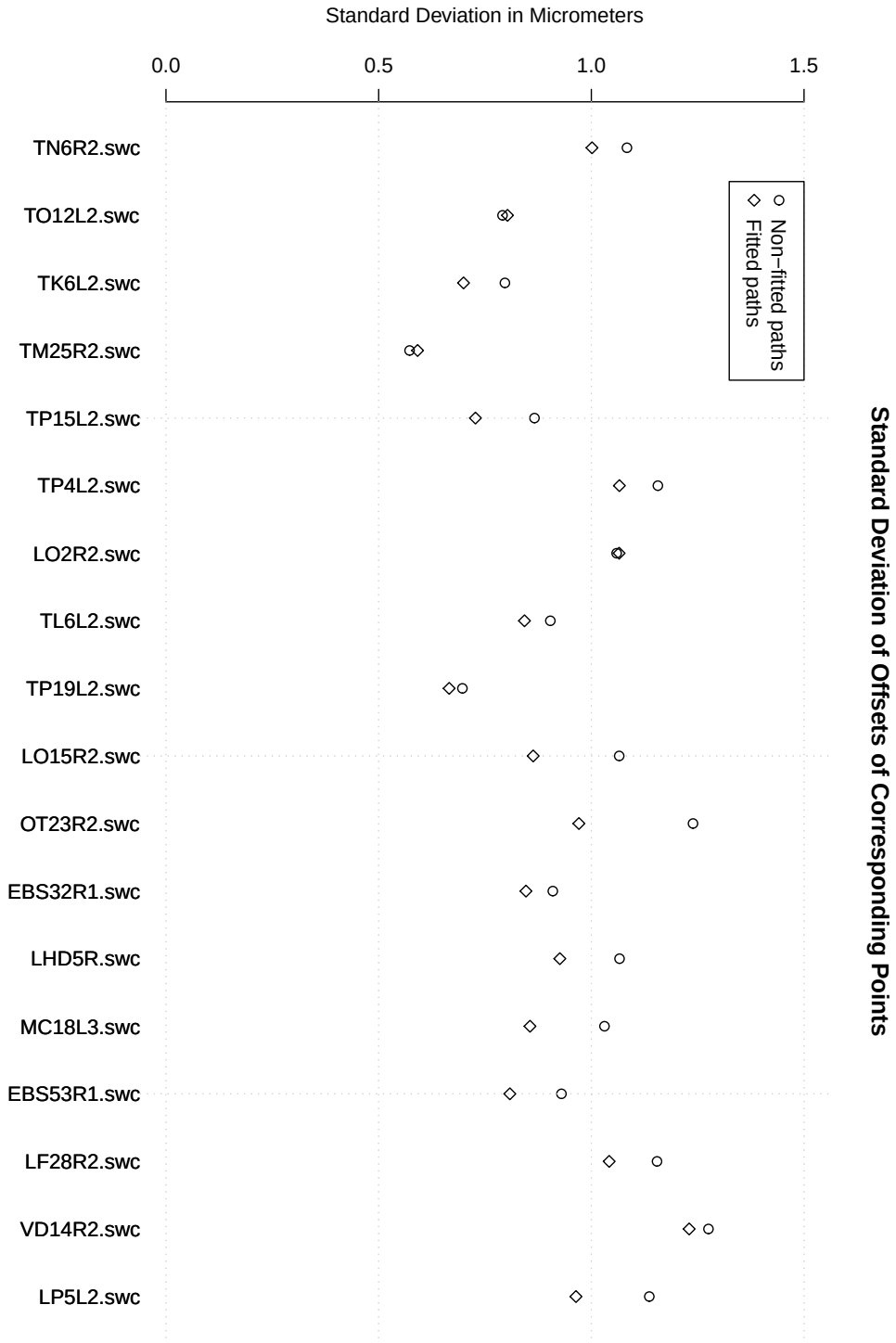


Figure 68 – A graph showing the standard deviation of offsets of points traced with Simple Neurite Tracer from those in the SWC files, evaluated for each point in the SWC file, once trying to find correspondences in the fitted paths and once in the non-fitted original paths.

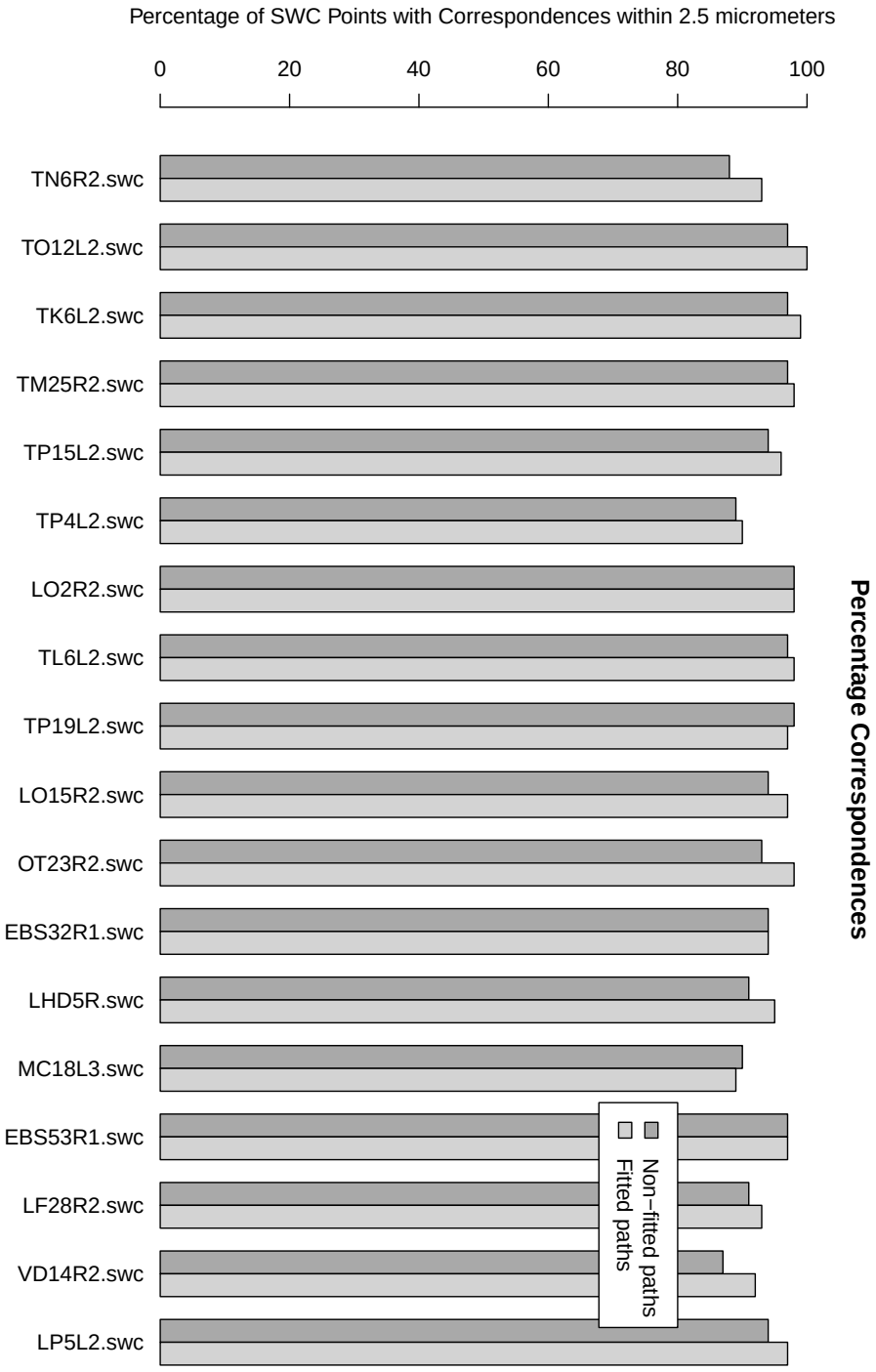


Figure 69 – A graph showing the percentage of points in each SWC file for which a corresponding point was found in the tracings generated with Simple Neurite Tracer. The darker bars show the case where correspondences were looked for in the unfitted paths, while the lighter bars show the percentage correspondence in the fitted paths.

Base Name	Paths	Total Length (µm)	Branch Points
71yABwestmost	35	2809	35
71yAN	18	1536	14
71yAM	*16*	*1583*	*11*
71yAF	*16*	*1486*	*11*
71yAS	21	2936	13
71yAAeastmost	17	2174	17
71yAT	25	2143	26
71yAR	9	989	7
71yAQ	*19*	*1587*	*11*
mhl-71yxUAS-lacZ(0)	*21*	*2035*	*16*

Base Name	Paths	Total Length (µm)	Branch Points
210y-40x-central-complex-CE	24	1832	12
210yAO	27	1917	17
210y-40x-central-complex-CD	30	2361	25
210yAP	12	1181	7
210yAC	33	3510	15
210yAE	24	2689	20
210y-40x-central-complex-CA	45	4116	30
210yAD	18	2050	16
210y-40x-central-complex-CB	*21*	*2021*	*22*
mhl-middle-ish(onlygoodoneon(E))210yxUAS-lacZ(0)	*32*	*2758*	*29*

Base Name	Paths	Total Length (µm)	Branch Points
c005BA	30	3198	24
c005BE	27	2959	16
c005BF	27	2099	18
c5xUAS-CD8GFP-40x-central-complex-BE	12	1136	4
c5xUAS-lacZ-40x-cc-BC	*23*	*2106*	*12*
c005BD	22	2370	10
c005BB	30	3544	25
c5xUAS-CD8GFP-40x-central-complex-BF	9	713	2
c5xUAS-lacZ-40x-cc-BA	*14*	*1927*	*7*
c005BC	*25*	*2160*	*17*
mhl-westmost(D)c5(0)	35	4584	17
c5xUAS-lacZ-40x-cc-BB	*17*	*1670*	*10*
c5xUAS-CD8GFP-24x-cc-BD	*15*	*1502*	*9*
mhl-middle(C)c5(0)	19	2501	11
c5xUAS-CD8GFP-40x-central-complex-BG	*14*	*1535*	*5*

Base Name	Paths	Total Length (µm)	Branch Points
c061AL	30	2418	31
c061AK	34	2873	32
c061AG	29	2222	33
c061AU	29	2193	24
c061AI()	29	1734	28
c061AV	11	1165	10
c061AJ	28	2190	24
c061AH	24	1749	26
mhl-theotherone(A)c61(0)	31	3503	13
mhl-eastmost(A)c61(0)	*49*	*3282*	*28*
mhl-westmost(B)c61(0)	32	3305	29
mhl-northernmost(A)c61(0)	23	2504	14

These data, representing paths and branch points for over 40 brains, present an interesting challenge

to analyse. The complexity of the tracings in a single brain can be seen in the exemplar images in Figures 88, 75, 70 and 81. The key question is what kinds of analysis will be interesting, given that in most cases we cannot unambiguously determine which neurons are distinct, and there is no easy automatic way to distinguish different regions of the neuron. In the case of each of the lines c005, c061 and 71y I have picked a known neuron type from [Hanesch et al., 1989] that can be found in those images to study the variation of. Even in these cases I am restricting the paths to the part that would be considered the primary process on the way to the fan-shaped body, ignoring branches that lead to separate arbors. In the case of 210y I have chosen to look at an apparent connection between the fan-shaped body and the mushroom body. In each of these case studies the methodology is similar for aggregating the data from multiple brains, and this is described in the next section.

5.5.1 Aggregating Data From Multiple Brains

A necessary initial step in the analysis was to transform all the traces files into the same co-ordinate space using the CMTK-generated elastic transformations described in Chapter 4. Since CMTK generates an elastic mapping from points in the template space to points in the model space, this meant calculating an inverse of the function for each image. I generated these with a computationally expensive, but algorithmically simple, nearest-neighbour approach. Unfortunately but inevitably, the inverse transformation may break up some paths or lose branch points when points lie outside the template space after transformation.

For each given line and neuron type, I loaded the traces into the Simple Neurite Tracer plugin and recorded any paths which appeared to be part of the neurons in question. These were recorded in a Ruby data structure, for example:

```
{ "71y left Fl neurons" =>
        { "71yABwestmost" =>
                    [ [ "Path (0)" ],
                      [ "Path (1)", "Path (7)" ] ],
          "71yAN" => [ [ "Path (1)" ],
                      [ "Path (16)", "Path (1)" ] ],
          "71yAM" => [ [ "Path (2)", "Path (0)"],
                      [ "Path (3)", "Path (2)", "Path (0)" ] ],
          "71yAS" => [ [ "Path (5)", "Path (0)" ],
                      [ "Path (6)", "Path (5)", "Path (0)" ] ],
```

```
"71yAR" => [ [ "Path (5)" ] ],
"71yAQ" => [ [ "Path (17)", "Path (0)" ],
             [ "Path (18)", "Path (0)" ] ],
"mhl-71yxUAS-lacZ(0)" =>
           [ [ "Path (12)", "Path (6)" ] ] }
```

I wrote a JRuby script for Fiji that loaded each traces file, picked out the named paths for a particular neuron type and added them to the 3D viewer with a distinct colour. This script also generated a key to these colours. Renderings of the aggregated traces can be seen in the subsequent sections, with neuropil regions added to provide context. The traces were added as paths with constant radius rather than the fitted versions, since when fitted paths were used, clearer paths with better reconstructions appeared to have greater weight and obscured other paths.

Examining these renderings by eye I picked out a "typical" neuron that would be used as the basis for calculating offsets to similar neurons in other brains. When calculating the offset from each point on this distinguished neuron to any point in another traces file, I used the same concept of a corresponding point as described in section 5.4 but with a much larger limit of within 50μm. This generally works well, but there are some unexpected spikes in the graph where the corresponding point jumps to a different path.

5.5.2 c005 Neurons

A typical trace from a c005 brain at various stages of reconstruction can be seen in Figure 70. Even in this relatively sparse GAL4 expression pattern, most of the network of processes is too complex to easily separate and identify into different neuron types. The most prominent types of neuron traced in the c005 images, however, are those leaving the frontal-superior edge of the fan-shaped body and proceeding occipitally to cell bodies lateral to the calyces of the mushroom bodies. They are known as Fl neurons ("Fan-shaped body laterally") in [Hanesch et al., 1989] to distinguish them from the other major type of large-field neurons that innervate the fan-shaped body, the Fm neurons ("Fan-shaped body median"), which pass through the centre of the ellipsoid body. I selected paths that appeared to make up neurons of this type for the left and right sides separately. These are shown in surface renderings in Figures 71 and 72. The typical deviations of registered neurons from other brains from the typical paths are shown in graph form in Figures 73 and 74.

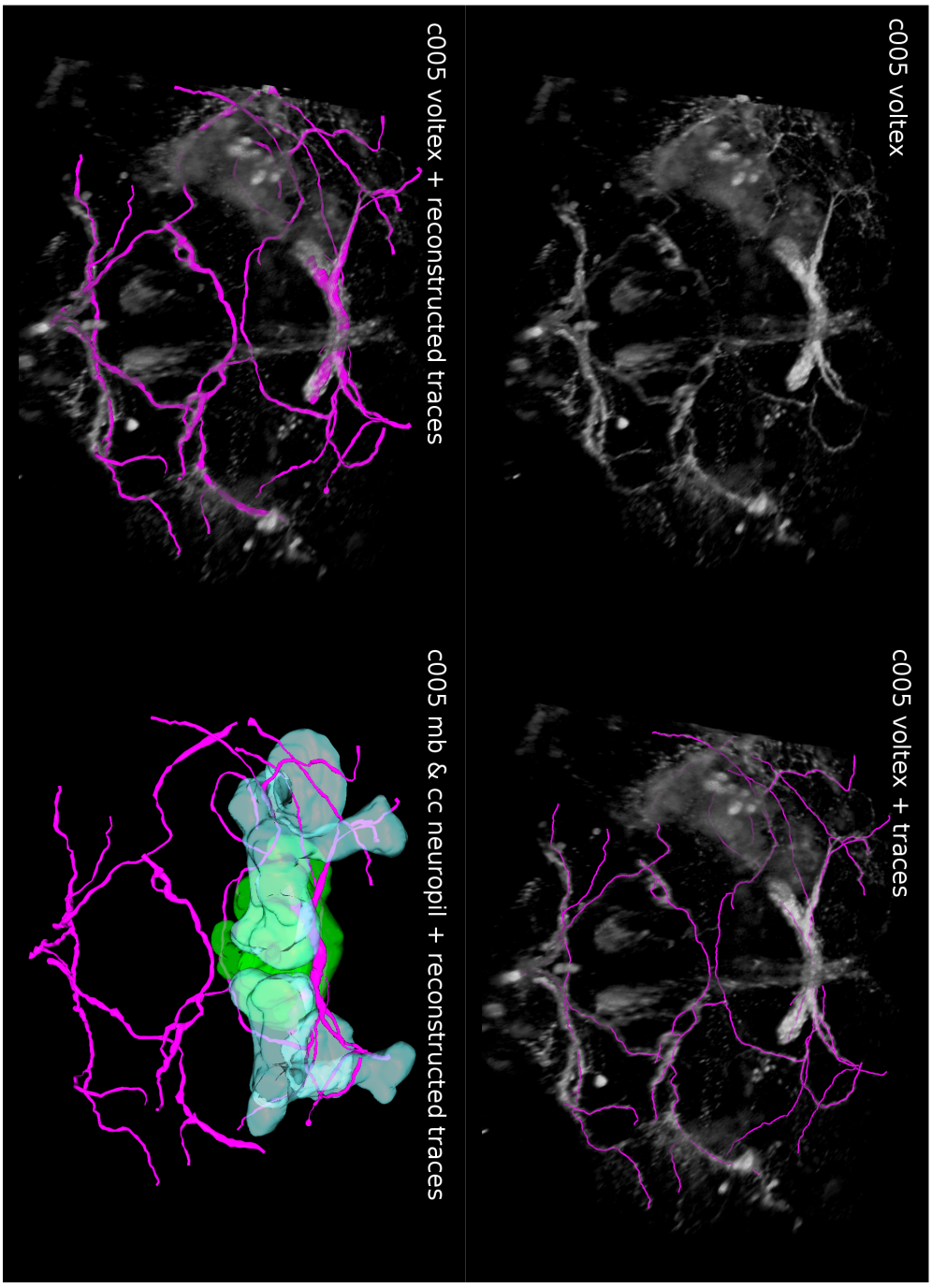


Figure 70 – Various stages of the tracing of an exemplar c005 brain ("c005BA").

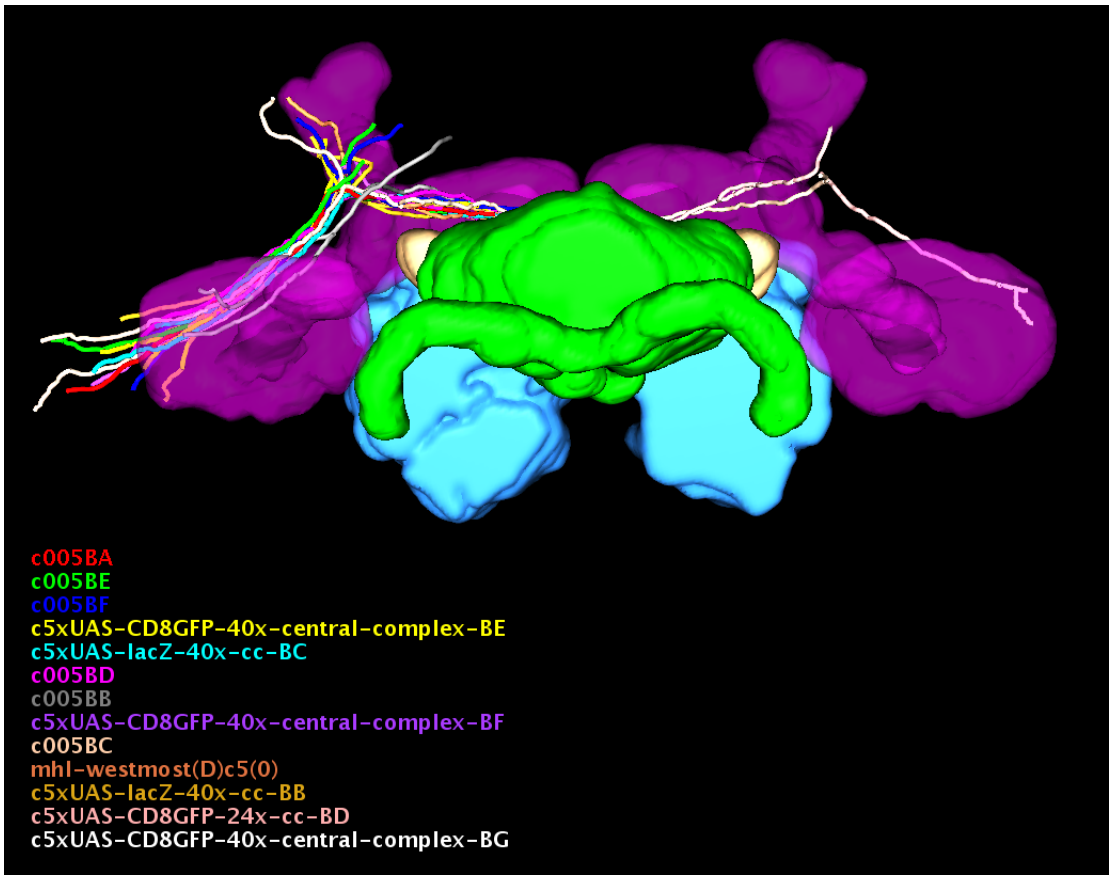


Figure 71 – This rendering shows registered c005 Fl neurons on the left, viewed from the back of the brain. In this and the subsequent renderings in this chapter, neuropil regions' surfaces have been added to give neuroanatomical context - these surfaces were generated from a labelling of the template brain. The surfaces have been made transparent in some cases for clarity. The neuron on the right hand is present since in the tracing process I followed these paths right across the brain. Currently Simple Neurite Tracer does not support a simple way to split paths after they have been traced.

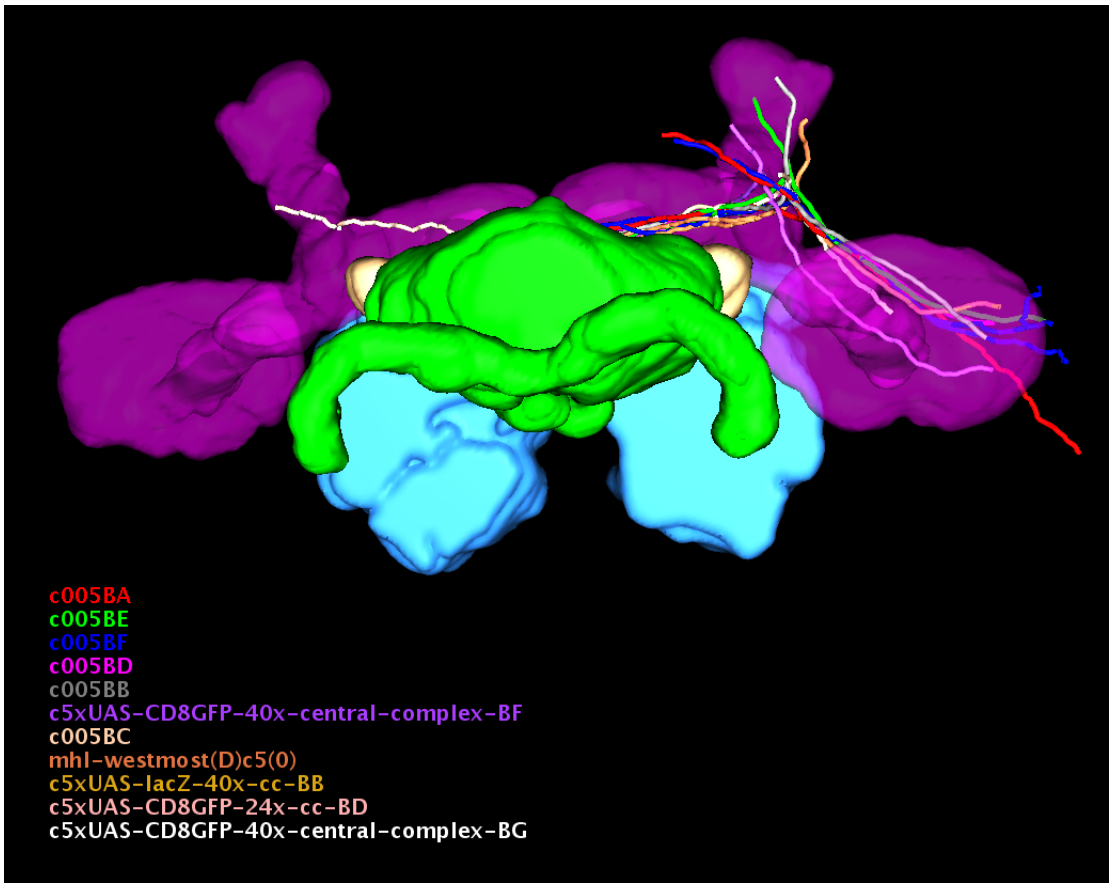


Figure 72 – This rendering shows registered c005 Fl neurons on the right, viewed from the back of the brain. The extra path on the left is present for the same reason as the extra paths in the previous image - while tracing I generated paths which passed through the centre of the brain and made up part of Fl neurons on both the left and the right.

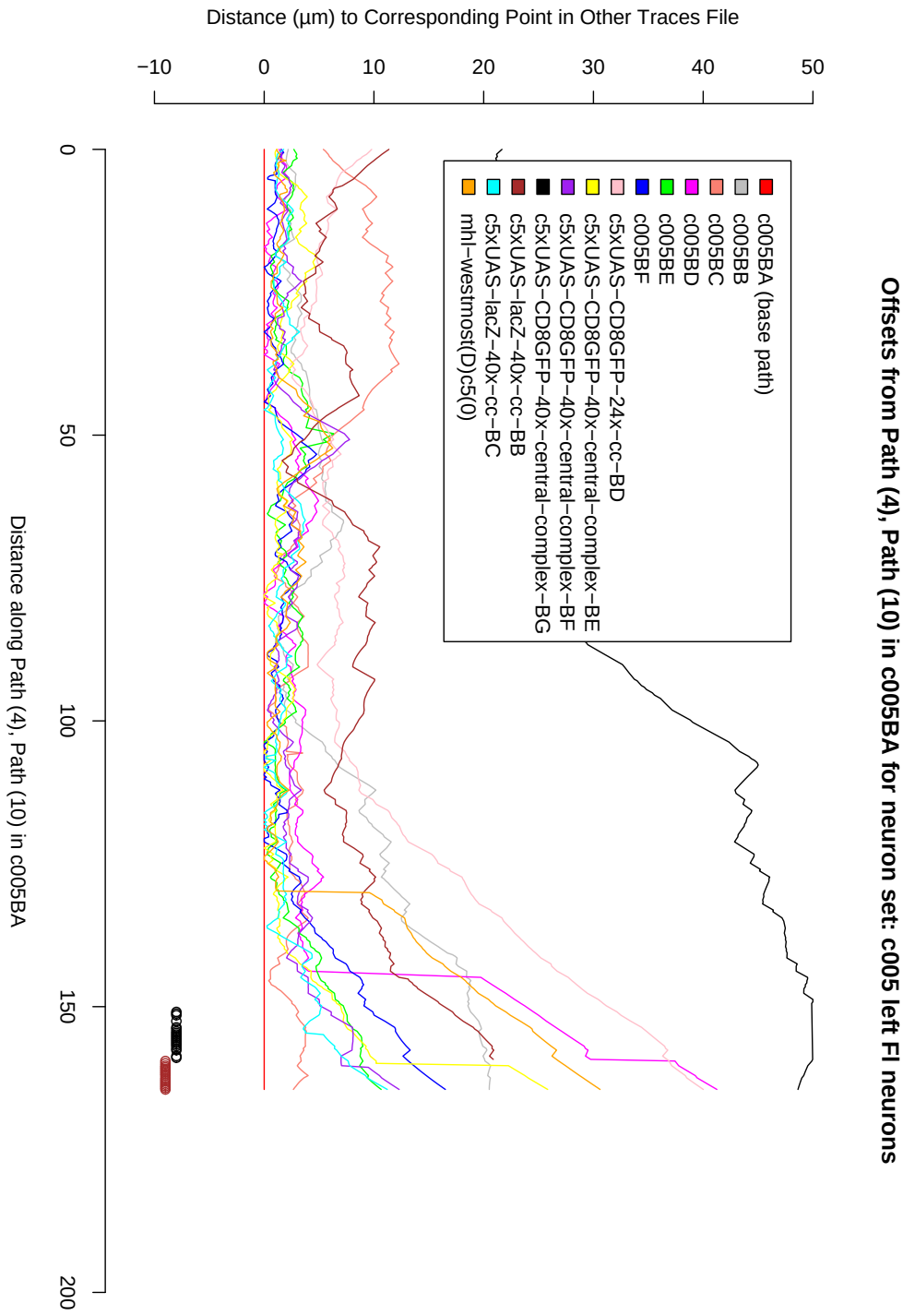


Figure 73 – A graph showing the distance to the nearest corresponding point in each other traces file to points along the typical c005 left Fl neuron. The colours of each line match those in Figure 71, although the badly misregistered path was removed from the rendering. The points marked in the negative y-axis region indicate where no corresponding points could be found in one of the other traces files. 0 on the x-axis corresponds to the end in the fan-shaped body, while the higher x values correspond to the lateral extent of the neurons.

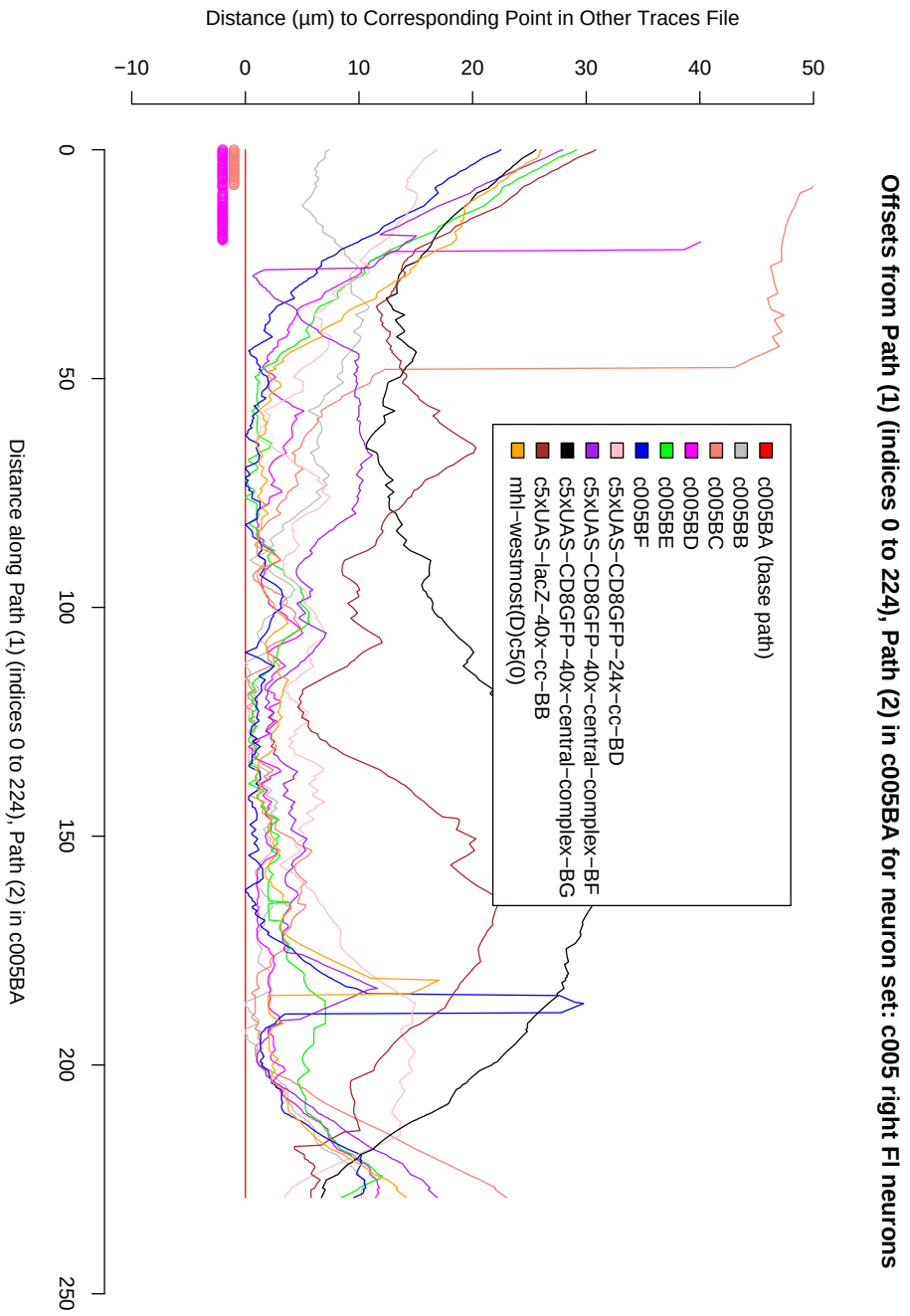


Figure 74 – A graph showing the distance to the nearest corresponding point in each other traces file to points along the typical c005 right Fl neuron. The colours of each line match those in Figure 72, although the two badly misregistered paths were removed from the rendering. The points marked in the negative y-axis region indicate where no corresponding points could be found in one of the other traces files. 0 on the x-axis corresponds to the most lateral extent of the neurons, while the higher x values correspond to the fan-shaped body ends. The paths from the file `c5xUAS-lacZ-40x-cc-BB` which can be clearly seen to be badly adrift in this graph were misregistered and so I removed them from the renderings in Figure 72.

5.5.3 210y Neurons

A typical trace from a 210y brain at various stages of reconstruction can be seen in Figure 75. A number of interesting features can be seen in such traced images, including:

1. A long horizontal tract behind the fan-shaped body.
2. A possible link between an inferior layer of the fan-shaped body and the gamma lobes of the mushroom body.
3. Fm neurons running from cell bodies around the calyces, through the ellipsoid body and to the inferior fan-shaped body.
4. Large flattened neurons throughout the mushroom body, in particular down the peduncles.
5. Some small-field neurons passing from the protocerebral bridge to other elements of the central complex.
6. A process running between the tips of the mushroom body gamma lobes.

As discussed in the introduction (section 1.3.3) the possibility of neurons that allow information to pass directly between the fan-shaped body and the mushroom body is an exciting one, so these are the paths that I have picked out in this results section. This link is only clearly traceable in 5 of the 210y images, so there are fewer images represented in these images and graphs than for the other lines. Figures 76 and 77 show the neurons as surface renderings.

As usual with these data, we have the problem that so many neurons are contained in each image that we cannot make definite statements about whether the process is part of a neuron that passes information from one region to another. Both the fan-shaped body and the gamma lobes of the mushroom body have strong expression, presumably from many different neurons, and there may well be no synaptic connections in either region. The expression pattern in the fan-shaped body, at least, does seem typical of the dendritic or synaptic arbors typically seen there.

Another reason to be cautious about this connection is that it is similar to a process that can be seen in the line NP6510, from Prof Kei Ito's lab.[54] The expression pattern of this GAL4 line is somewhat more sparse than 210y, and we can mark out a couple of possible complete paths from cell bodies

[54] I am very grateful to Dr Joanna Young for pointing out this detail of that line to me.

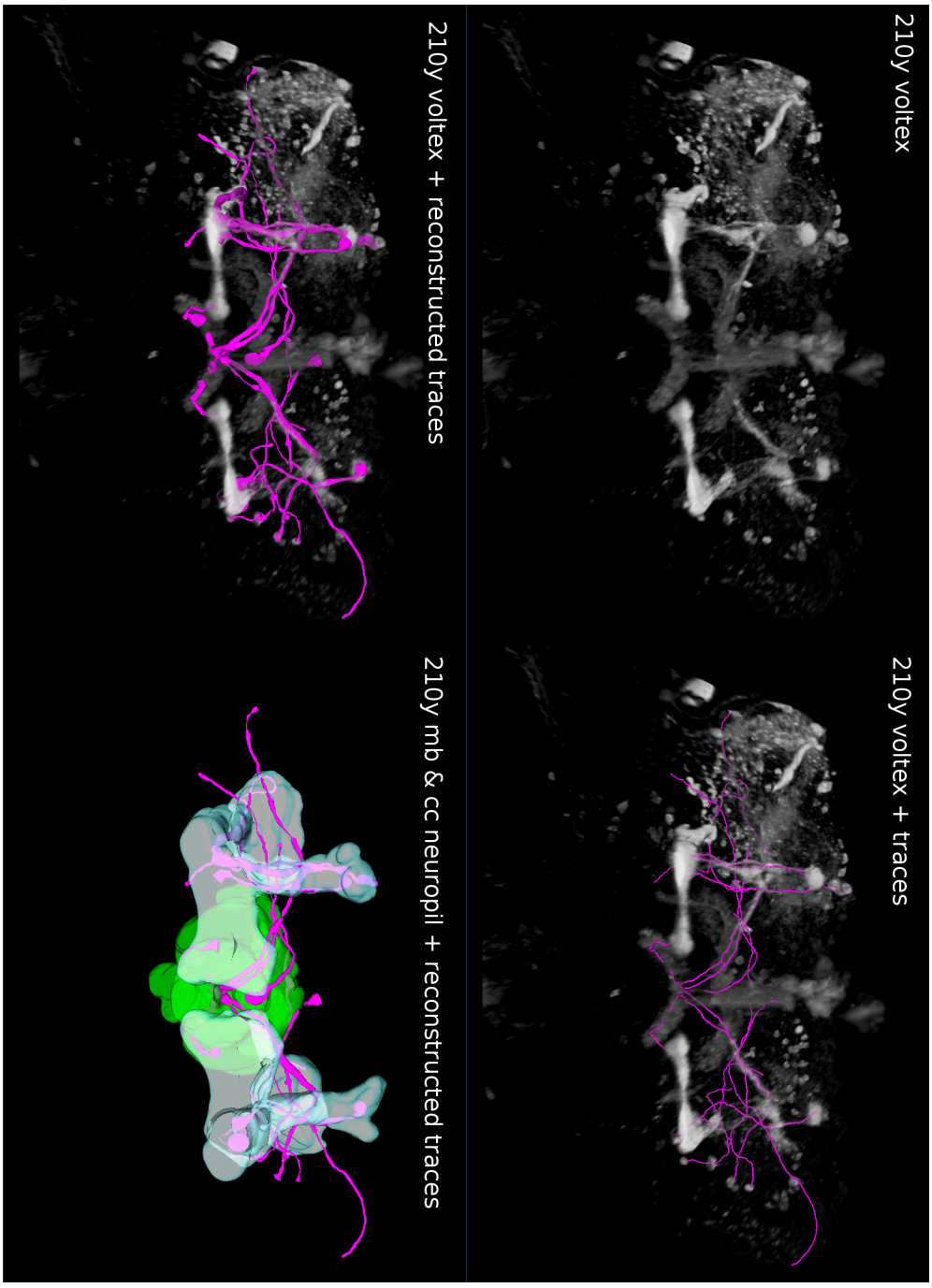


Figure 75 – Various stages of the tracing of a 210y brain ("210y-40x-central-complex-CD").

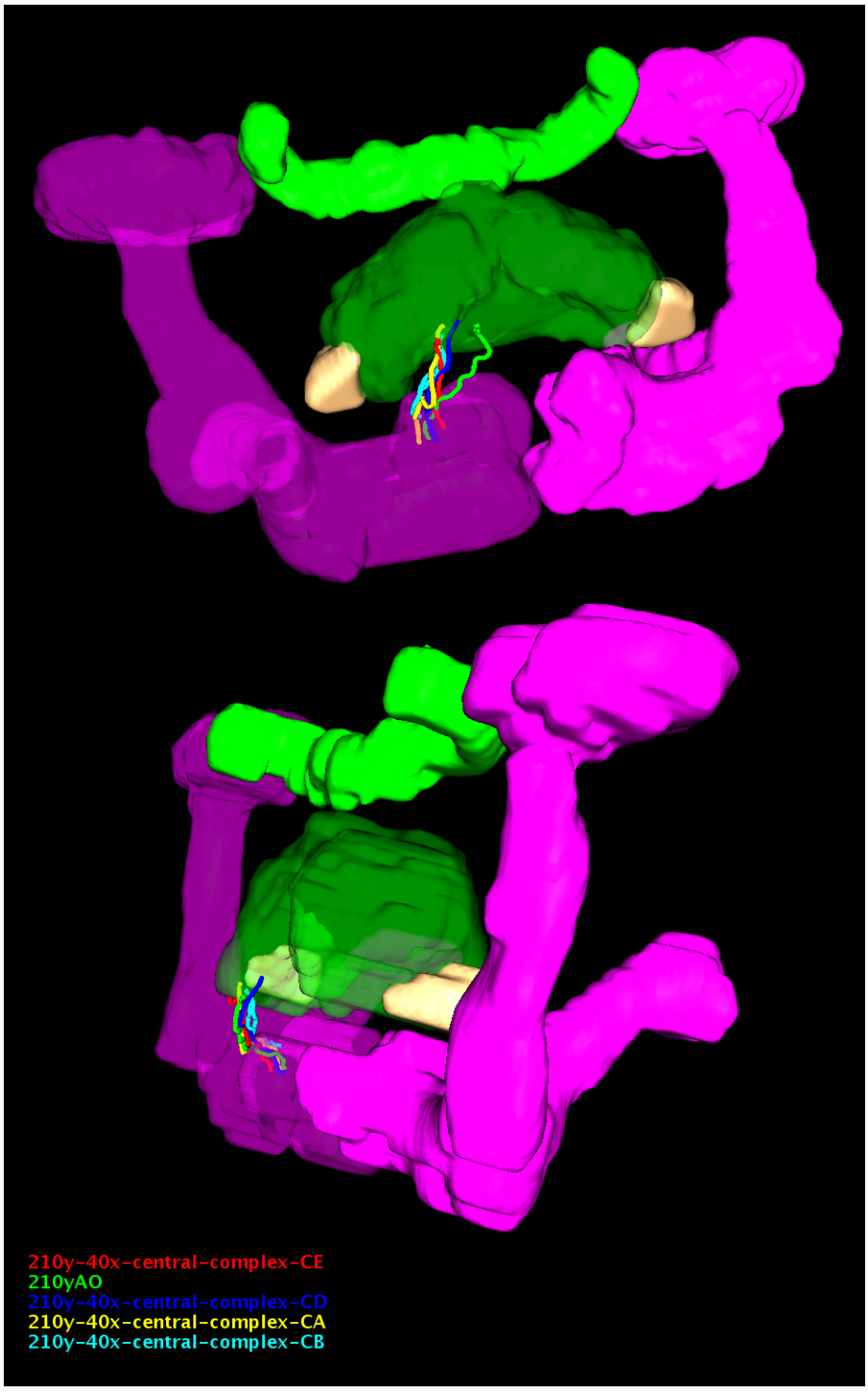


Figure 76 – Registered 210y neural processes passing between the fan-shaped body and the mushroom body on the left, shown from two different angles - the top view is from directly below the brain, as if looking along the body axis, while the bottom is a similar view from a more lateral position.

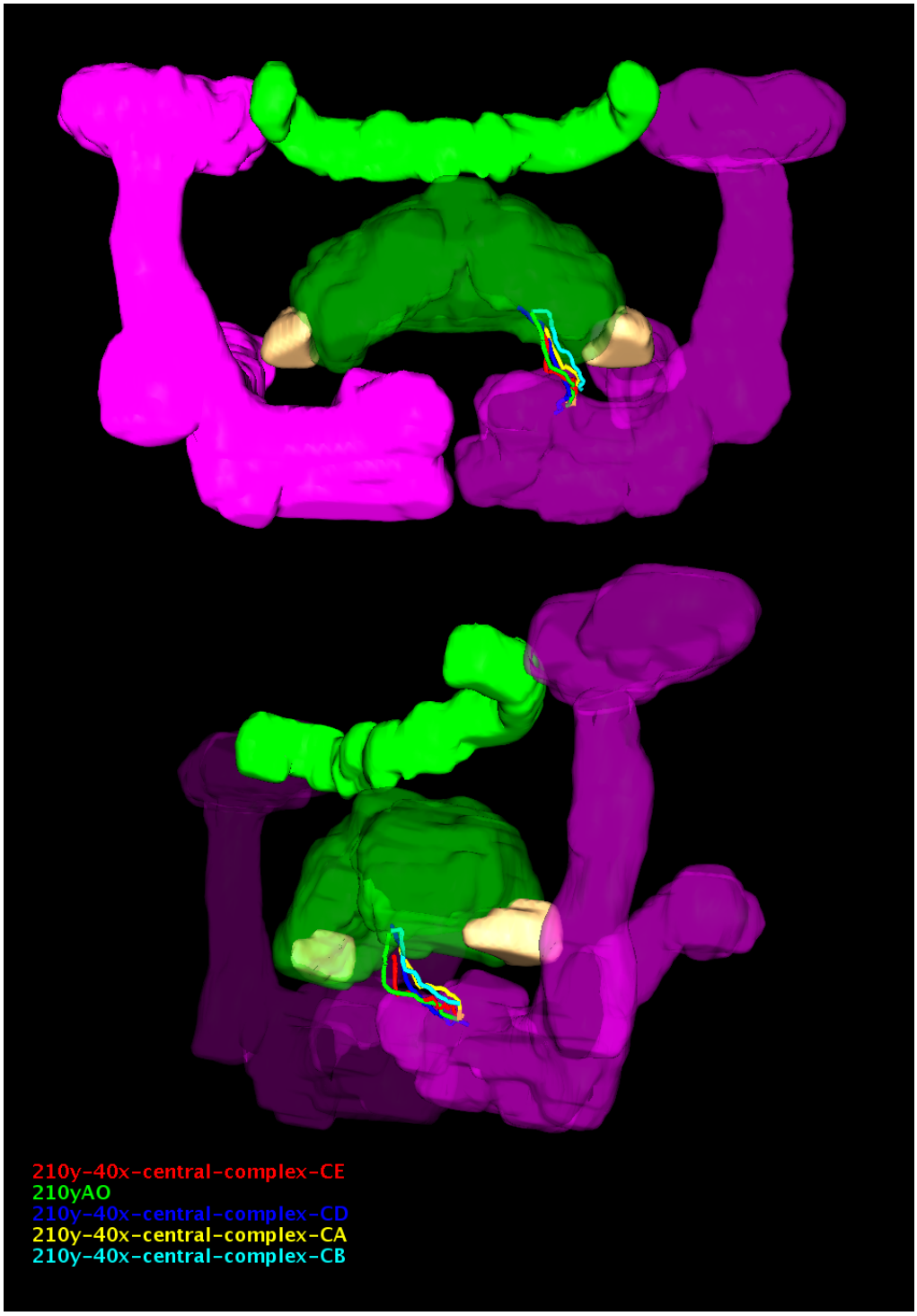


Figure 77 – Registered 210y neural processes passing between the fan-shaped body and the mushroom body on the right, shown from two different angles - the top view is from directly below the brain, as if looking along the body axis, while the bottom is a similar view from a more lateral position.

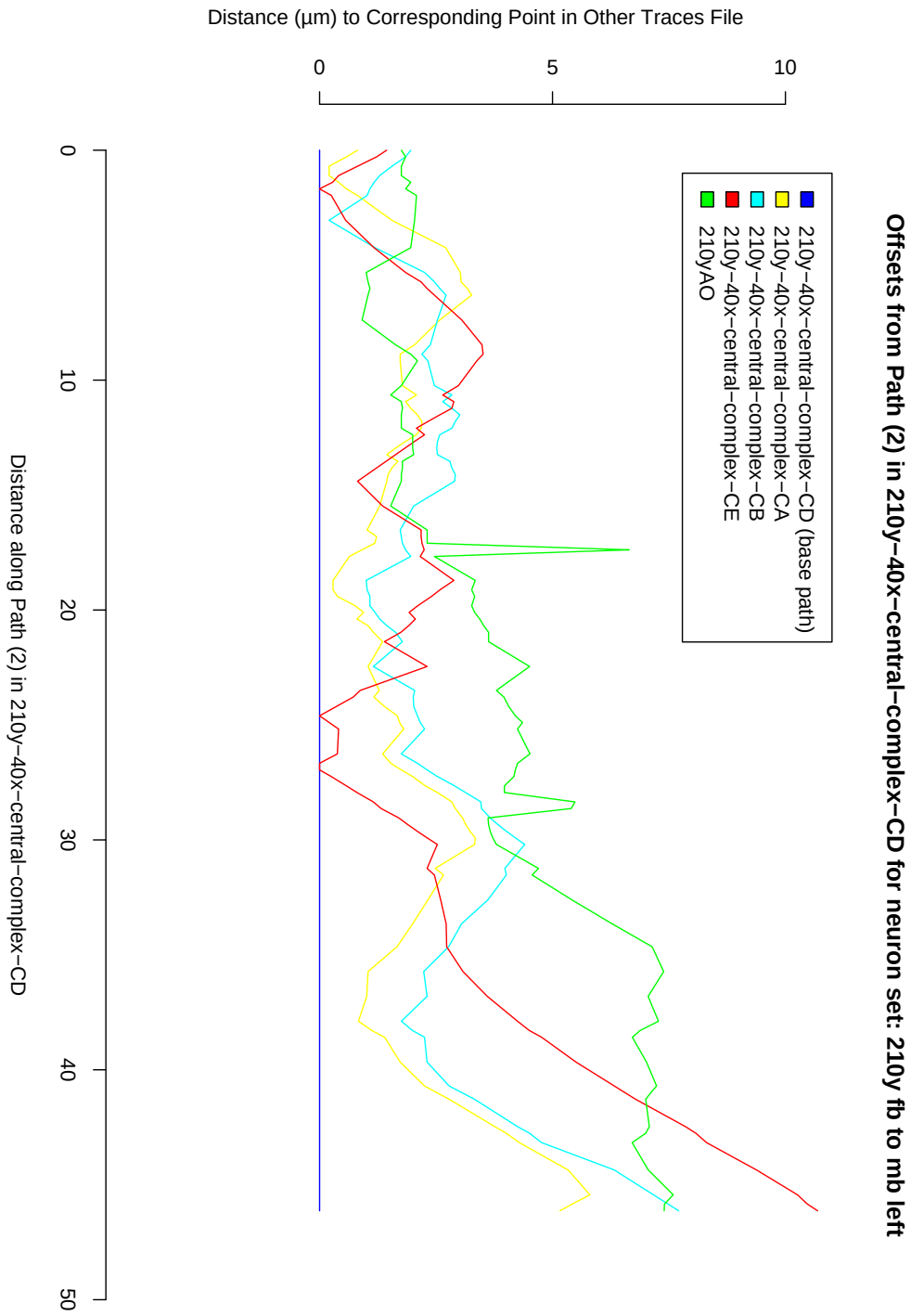


Figure 78 – A graph showing the distance to the nearest corresponding point in each other traces file to points along the typical 210y fb to mb path on the left. The colours of each line match those in Figure 76. In this graph 0 on the x-axis corresponds to the mushroom body end, while the higher x values correspond to the fan-shaped body end.

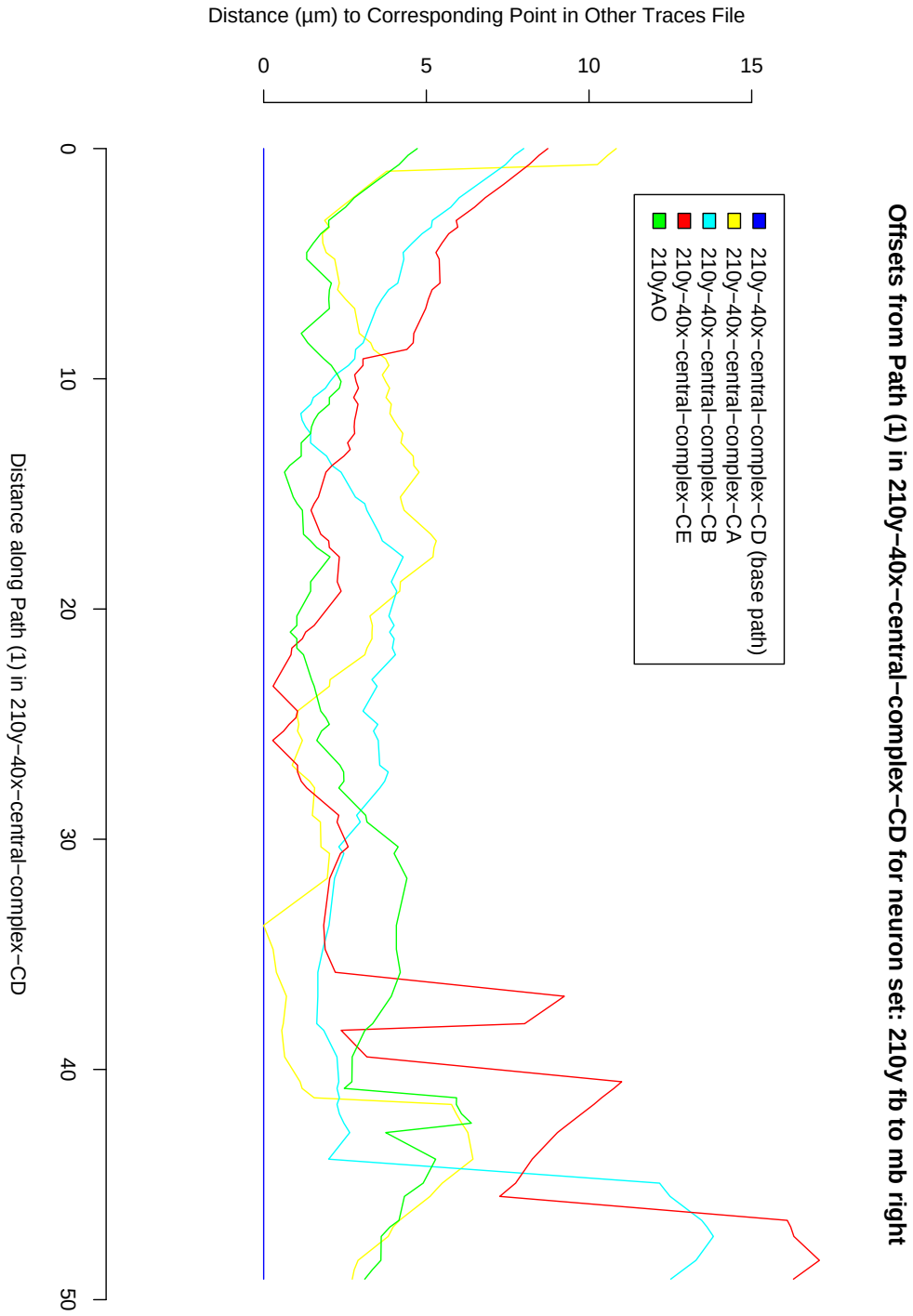


Figure 79 – A graph showing the distance to the nearest corresponding point in each other traces file to points along the typical 210y fb to mb path on the right. The colours of each line match those in Figure 77. In this graph 0 on the x-axis corresponds to the mushroom body end, while the higher x values correspond to the fan-shaped body end.

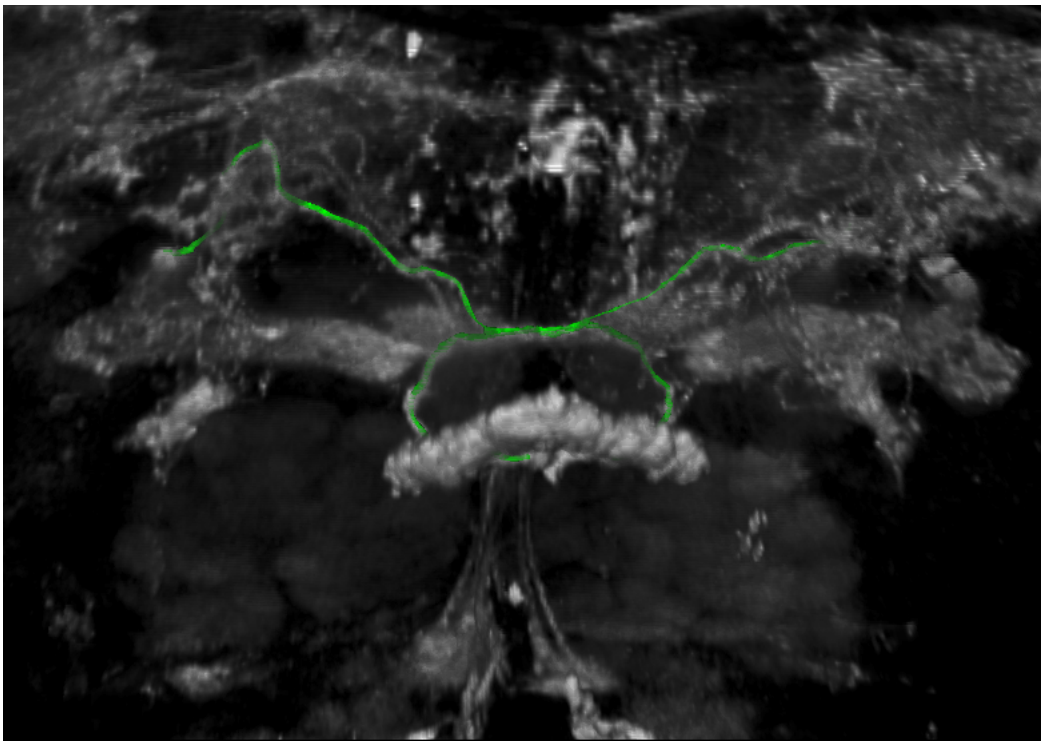


Figure 80 – This image is based on a scan of an adult fly brain kindly provided by Dr. Joanna Young. I added some traces to this image using Simple Neurite Tracer to indicate neuronal processes that seem to clearly include a similar section to the fan-shaped body to mushroom body link which can be seen in some of my 210y scans.

to the fan-shaped body in these images. A couple of these traces which include the similar section between the fan-shaped body and the mushroom body can be seen in Figure 80. This image, due to the selective nature of my path annotation, looks more promising than perhaps it should - the expression pattern of N6510 was studied in depth in the paper [Li et al., 2009], which included single neuron images generated with the FLP-out technique. These seem to show that this process does not arborize in the mushroom body, but instead in the lateral accessory lobe from a branch much nearer to the soma.

5.5.4 c061 Neurons

A typical trace from a c061 brain at various stages of reconstruction can be seen in Figure 81. The most obviously identifiable neurons here are the Fm neurons identified in [Hanesch et al., 1989] leading

from the fan-shaped body through the ellipsoid body canal and back to the posterior ventrolateral protocerebrum. Aggregated data for these paths are shown as in previous sections in Figures 82, 83, 84 and 85.

An interesting aspect of the traces in the c061 images is that I cannot find a convincing link between the fan-shaped body and the mushroom body in these traces, as we initially believed existed, i.e. any connections between the two are indirect.

5.5.5 71y Neurons

A typical trace from a 71y brain at various stages of reconstruction can be seen in Figure 88. The most prominent feature of the 71y expression pattern is very similar to that in c005: Fl neurons with cell bodies lateral to the calyces. 71y picks out a smaller set of neurons, however. Registered paths from Fl neurons from 71y traces files are shown in Figures 89 and 90. The former image also includes a transverse view of the brain which shows an interesting artefact with this tracing method: where the neurons enter the fan-shaped body neuropil at the frontal-superior edge of the fan-shaped body there is a dense chiasma where many neurons cross - currently there is no way in the tool to mark out such structures since they are not linear in shape, so I typically chose to end paths when they reach such a region. However, in the case of 71y (and c005) this region is actually outside the neuropil region as labelled from the nc82 scan. There are a variety of possible ad-hoc solutions to this problem: for example, Simple Neurite Tracer allows one to load a labels file to use to give contextual information about the current neuropil region in ImageJA's status bar - this allows the user to continue the path until it reaches the neuropil region they believe the path should be associated with. However, this is an unsatisfying solution, since once within a uniform region of high expression, the linear annotation of paths is obviously artificial.

My preferred solution to such problems is to build a classifier which at any point in an image stack will characterize the type of expression in that region into one of a number of categories - large regions of consistently high expression would be one of these, and these could be regarded as providing a connection to any neuropil region with which they overlap.

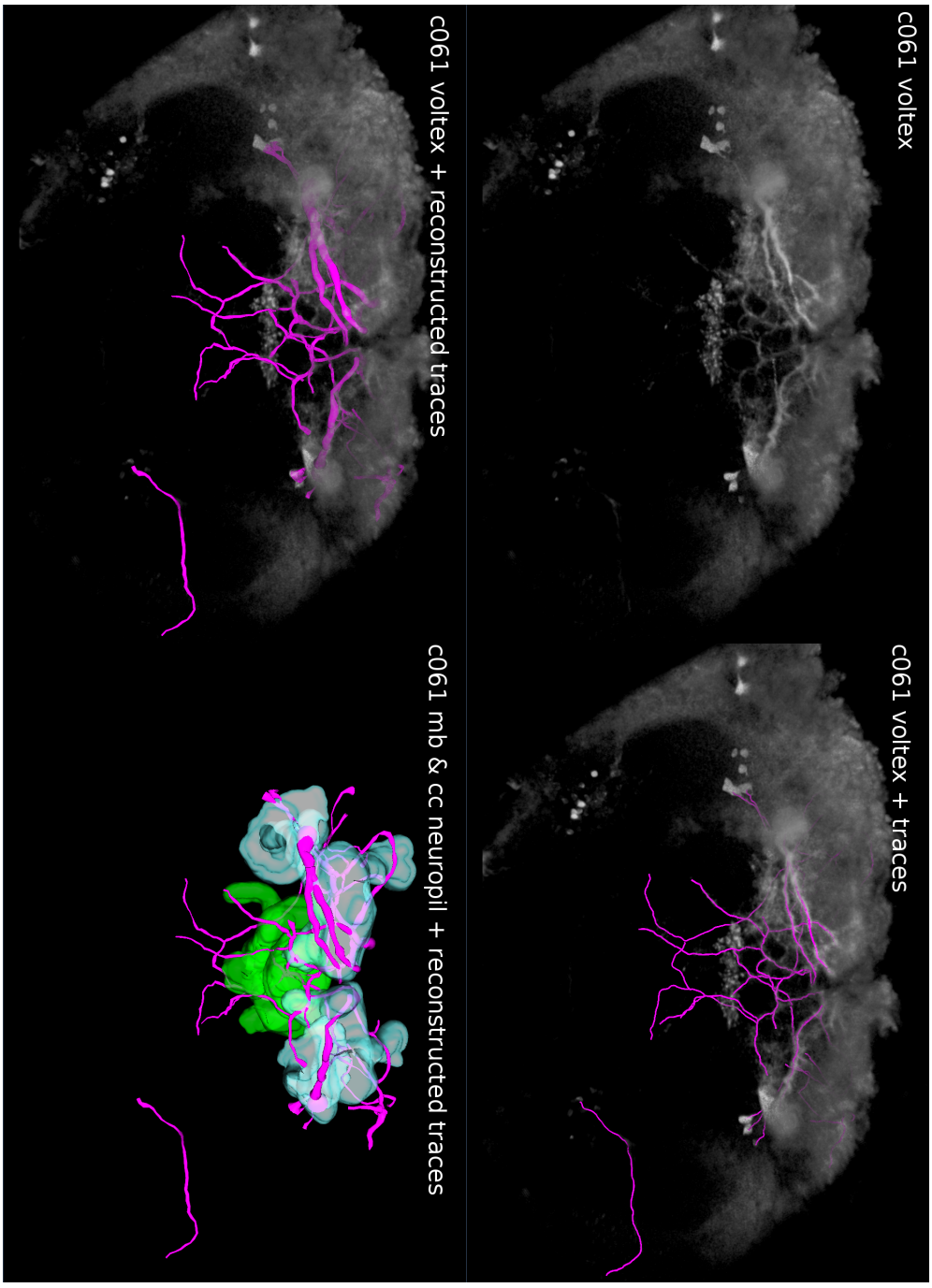


Figure 81 – Various stages of the tracing of a c061 brain (“c061AK”)

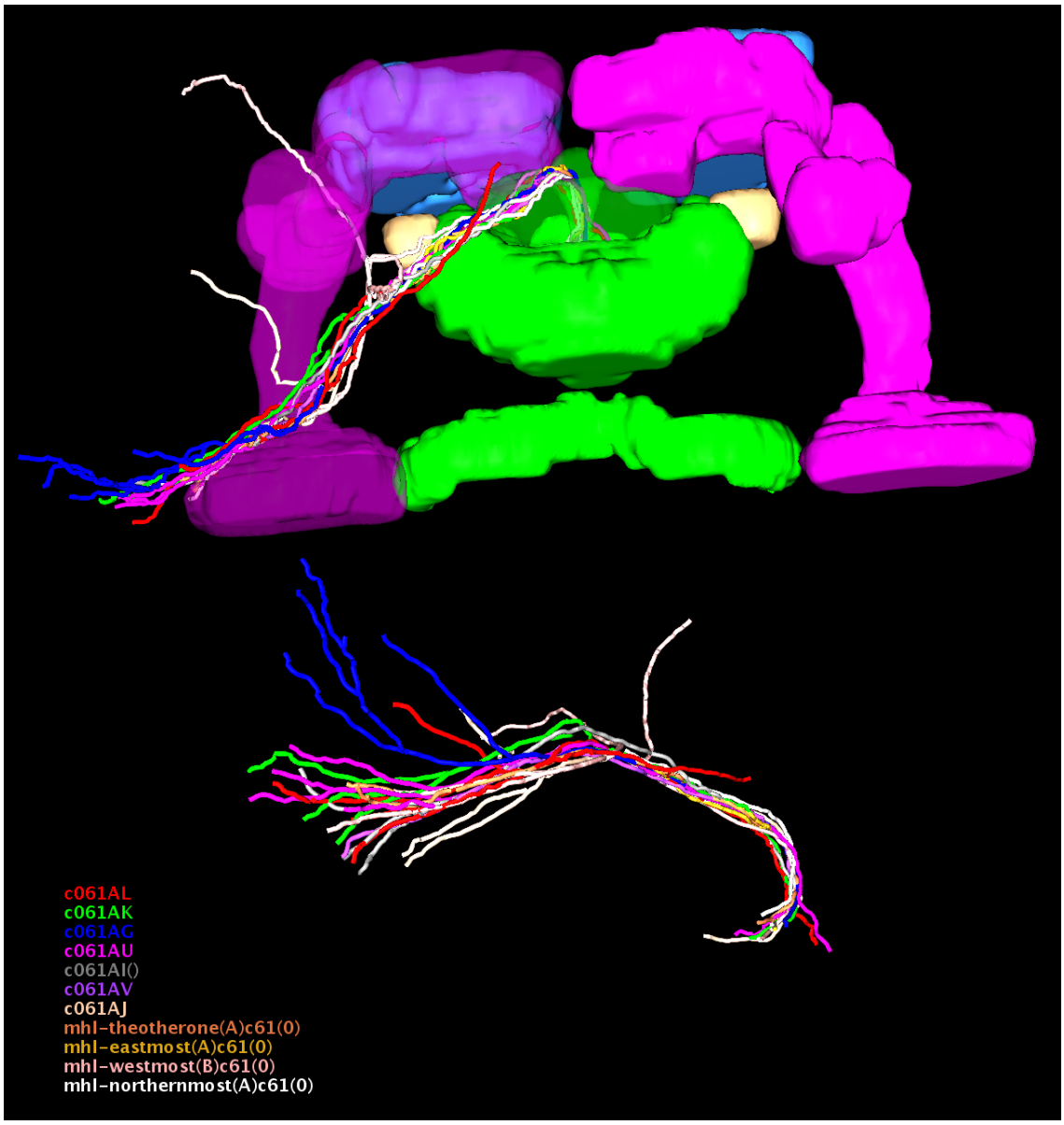


Figure 82 – Registered c061 traces that make up Fm neurons on the left side of the brain. The top view is looking down onto the top of the brain along the alpha lobes of the mushroom body from the template brain. The lower view shows only the neurons from all the different c061 brains, but from a more coronal view. (This shows that the cell bodies at the end of the traced paths in c061AG are offset from the others, which is not clear from the top view.)

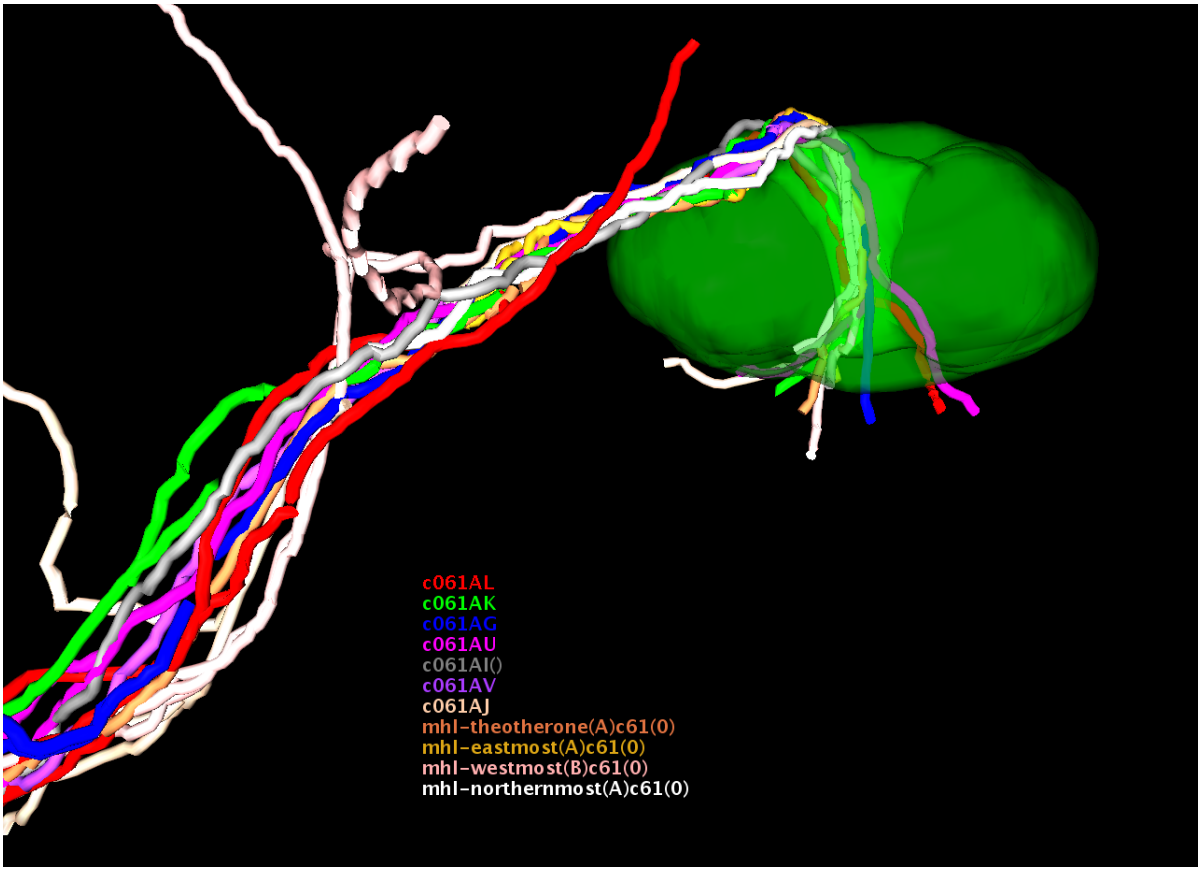


Figure 83 – Registered c061 traces that make up Fm neurons on the left side of the brain. This detail has only the ellipsoid body neuropil surfaces from the template brain added in a transparent rendering to clearly show the processes passing through the ellipsoid body canal.

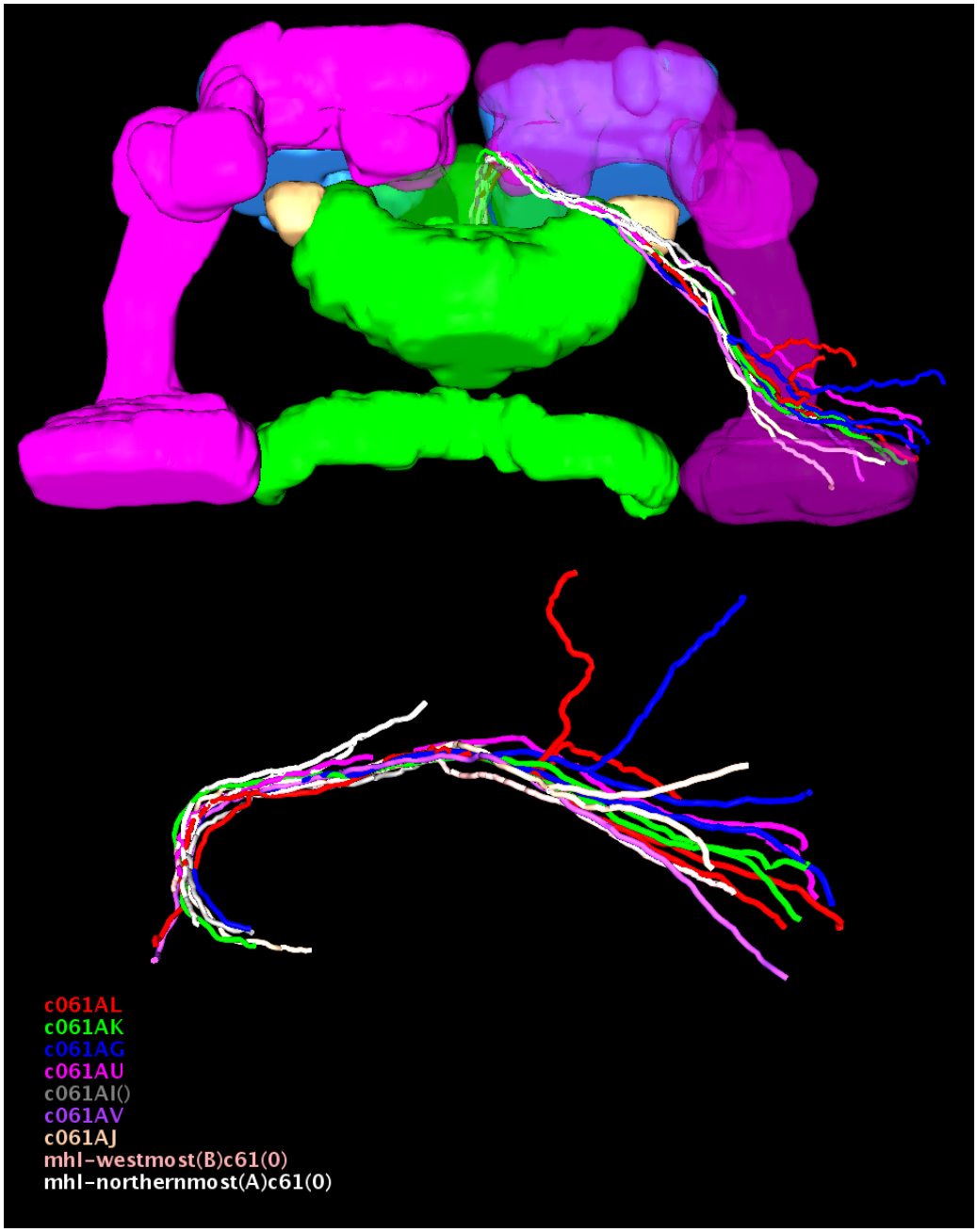


Figure 84 – Registered c061 traces that make up Fm neurons on the right side of the brain. The top view is looking down onto the top of the brain along the alpha lobes of the mushroom body from the template brain. The lower view shows only the neurons from all the different c061 brains, but from a more coronal view.

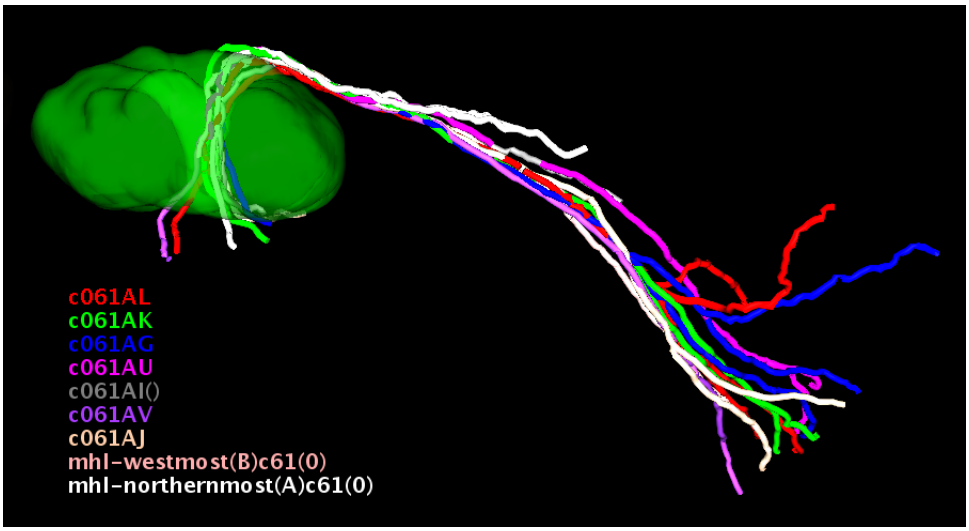


Figure 85 – Registered c061 traces that make up Fm neurons on the right side of the brain. This detail has only the ellipsoid body neuropil surfaces from the template brain added in a transparent rendering to clearly show the processes passing through the ellipsoid body canal.

5.6 Development (Auto Tracer)

The chief problem with the semi-automated Simple Neurite Tracer tool is that tracing out complex neuronal networks is very time-consuming and the results depend a great deal on the care which the user is prepared to take over the annotation. This is particularly the case when dealing with dense neuronal arbors. To address these problems I created a plugin which is designed to trace a brain thoroughly without user interaction after picking some initial parameters.

The user should first use the Tubeness plugin to generate a version of the image stack that emphasizes tube-like features at an appropriate scale and save this image as `[basename].tubes.tif`, where `[basename]` is the original basename of the image file. The performance of the plugin is affected by a number of parameters, but two of these must be chosen by the user based on this image. The first is the threshold m in this preprocessed image, above which all pixels are likely to be part of a path - these points of "high tubeness" are used both as start and goal points (called "seed points" below) in a series of best-first searches using Dijkstra's algorithm. m should be chosen such that almost all of the values above that threshold are part of neuron-like structures while at the same time it is not set so low that there are clear paths along the neurons of interest. Scattered points are fine, so the aim should be to minimize "false positives": points above the threshold that are not plausibly part of any

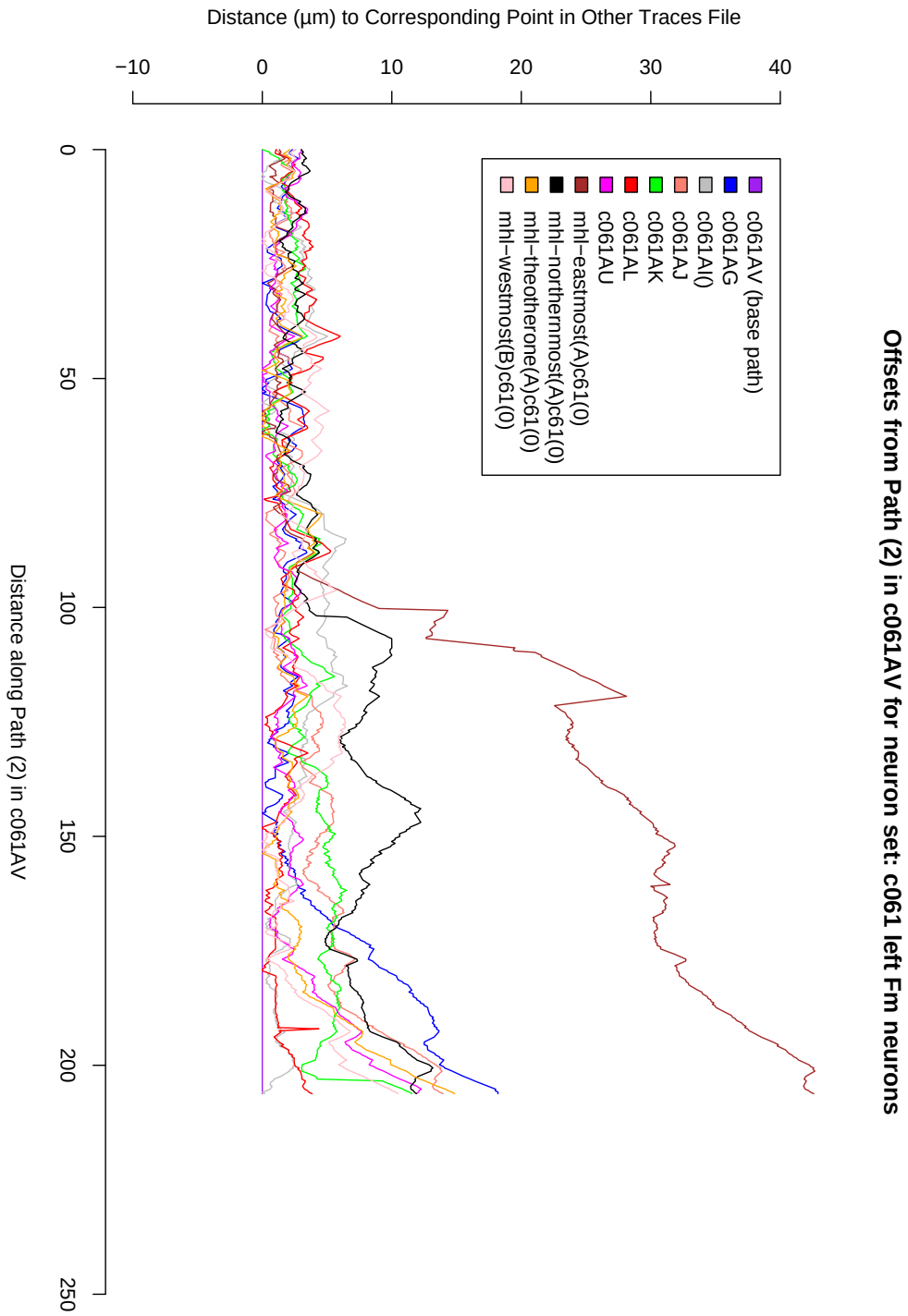


Figure 86 – A graph showing the distance to the nearest corresponding point in each other traces file to points along the typical c061 Fm neuron on the left. The colours of each line match those in Figures 82 and 83. In this graph 0 on the x-axis corresponds to the fan-shaped body end, while the higher x values correspond to the lateral ends.

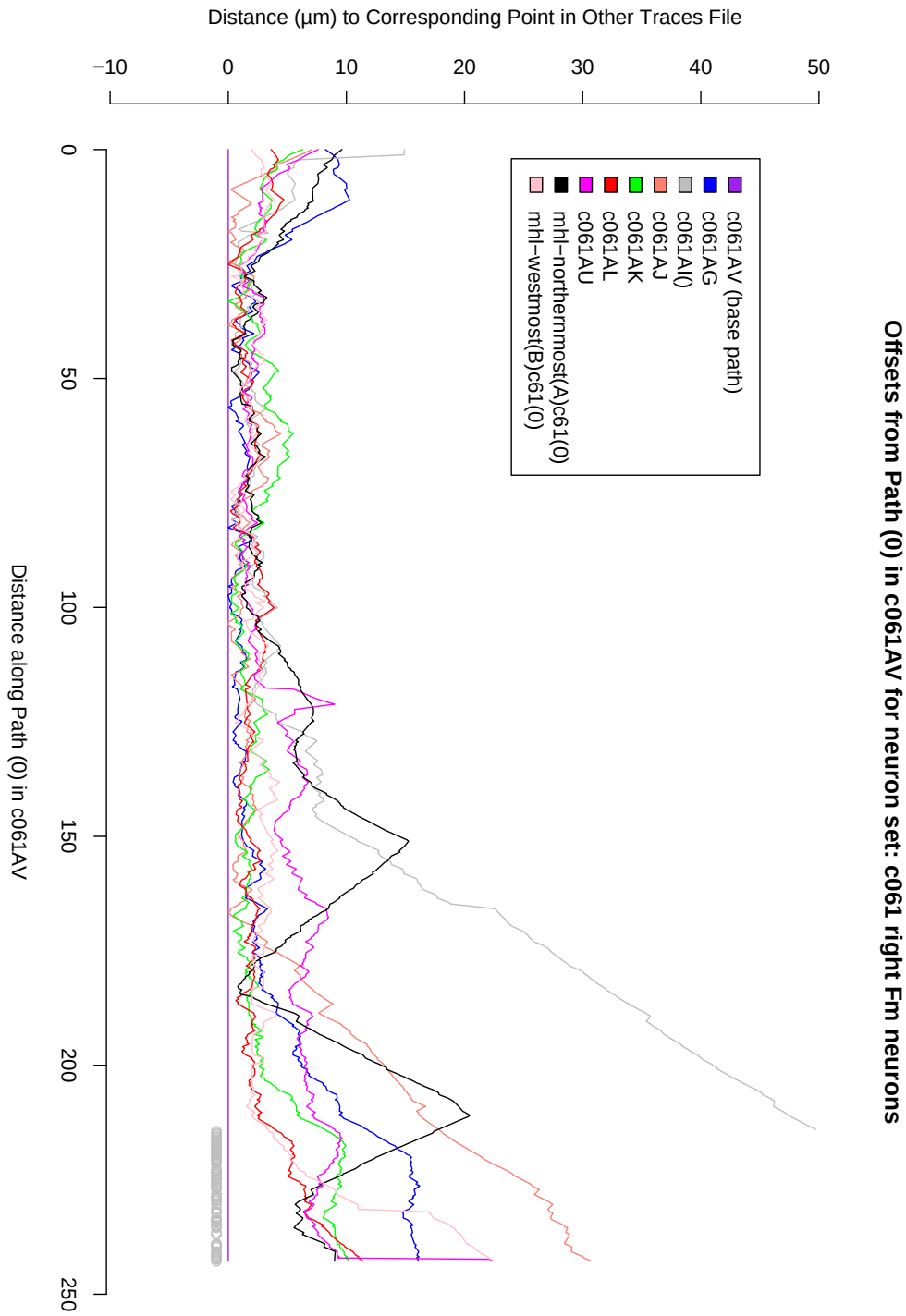


Figure 87 – A graph showing the distance to the nearest corresponding point in each other traces file to points along the typical c061 Fm neuron on the left. The colours of each line match those in Figures 82 and 83. In this graph 0 on the x-axis corresponds to the fan-shaped body end, while the higher x values correspond to lateral end.

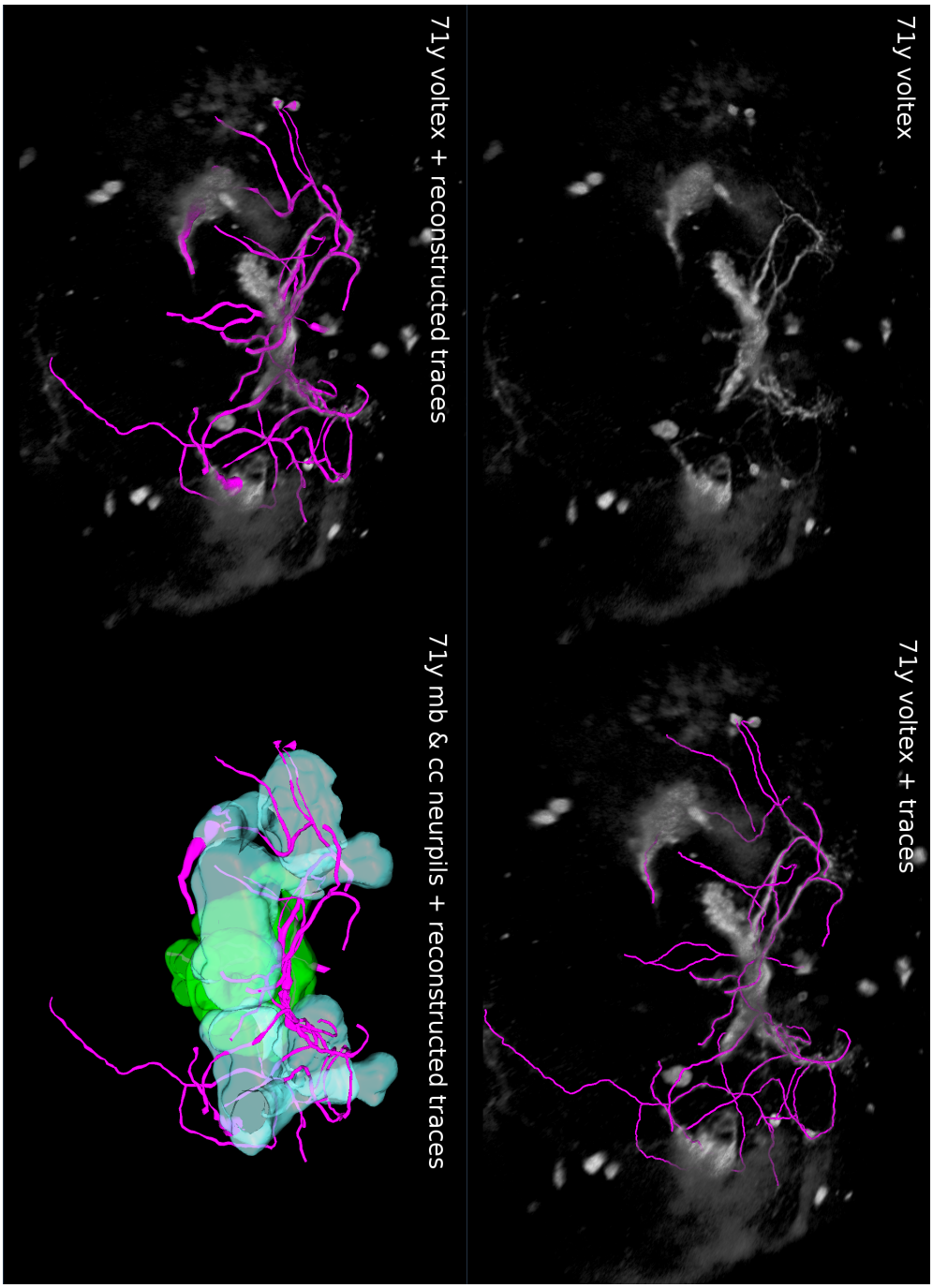


Figure 88 – Various stages of the tracing of a 71y brain ("71yABwestmost").

Figure 89 – Registered 71y Fl neurons on the left. The top view is from the occipital side of the brain, while the lower view is looking directly down along to the brain, approximately along the alpha lobes of the mushroom bodies. This latter view shows an interesting artefact of the tracing process, in that some of the traces stop short of the fan-shaped body neuropil. (This is discussed further in section 5.5.5.)

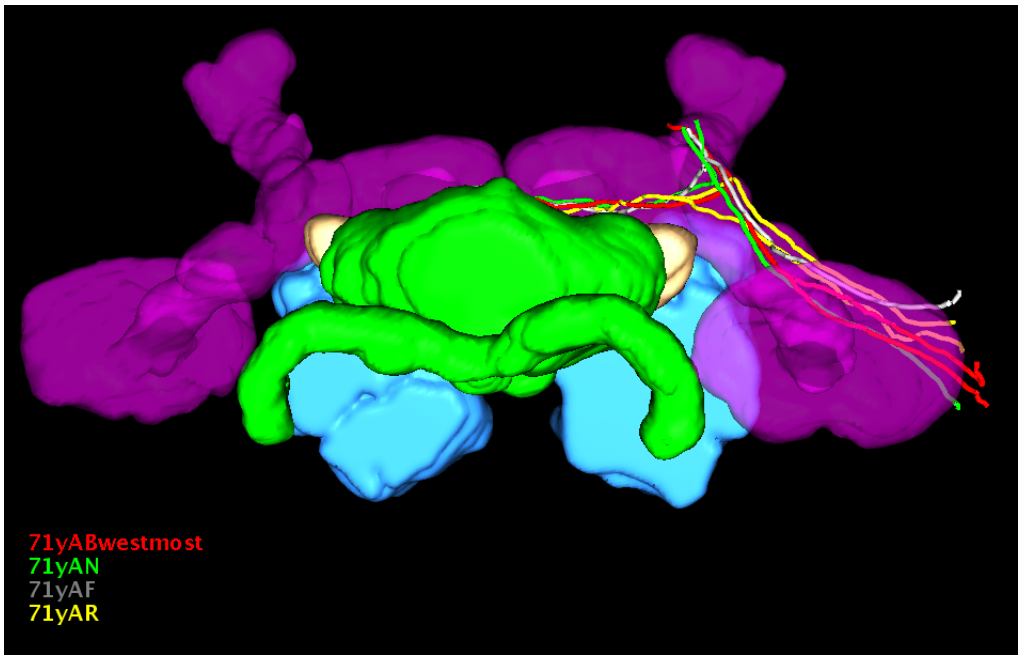


Figure 90 – Registered 71y Fl neurons on the right. This view is from the occipital side of the brain.

neuron-like structure. The second user-selected parameter is r, which specifies the minimum average tubeness allowed in a sliding window along any candidate path. This should be a lower value, which includes most neurons in the image when applied as a threshold to the tubeness image. Examples of reasonable choices for these values for one slice of an image are shown in Figure 93.

The other parameters which the plugin relies on, but which the user generally will not alter, are:

- The maximum time allowed for each search.
- The maximum number of iterations allowed for each search.
- The length of path segment used to calculate "mean tubeness" (i.e. the size of the sliding window).

The algorithm then proceeds as follows:

- All the points with tubeness values above m are put into a priority queue P, such that the most tube-like points are the first to be removed from the queue.
- While there are still points in P, we do the following loop:

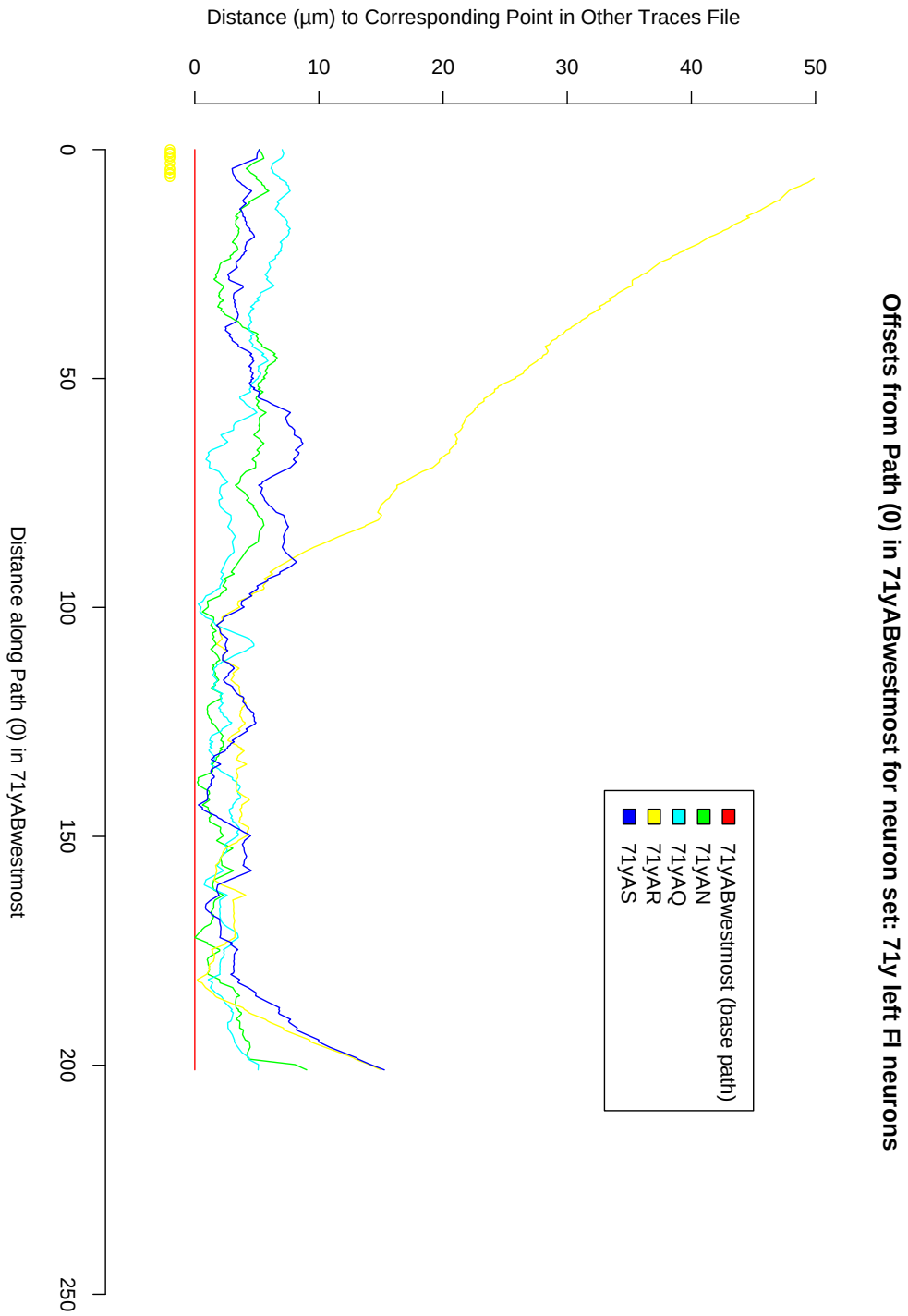


Figure 91 – A graph showing the distance to the nearest corresponding point in each other traces file to points along the typical c005 left Fl neuron. The colours of each line match those in Figure 89. The points marked in the negative y-axis region indicate where no corresponding points could be found in one of the other traces files. 0 on the x-axis corresponds to the end at the lateral extent of the neurons, while the higher x values correspond to points nearer to the fan-shaped body. The deviation of the labelled branch in the 71yAP (yellow) neuron from the others is visible in the rendered image above as well.

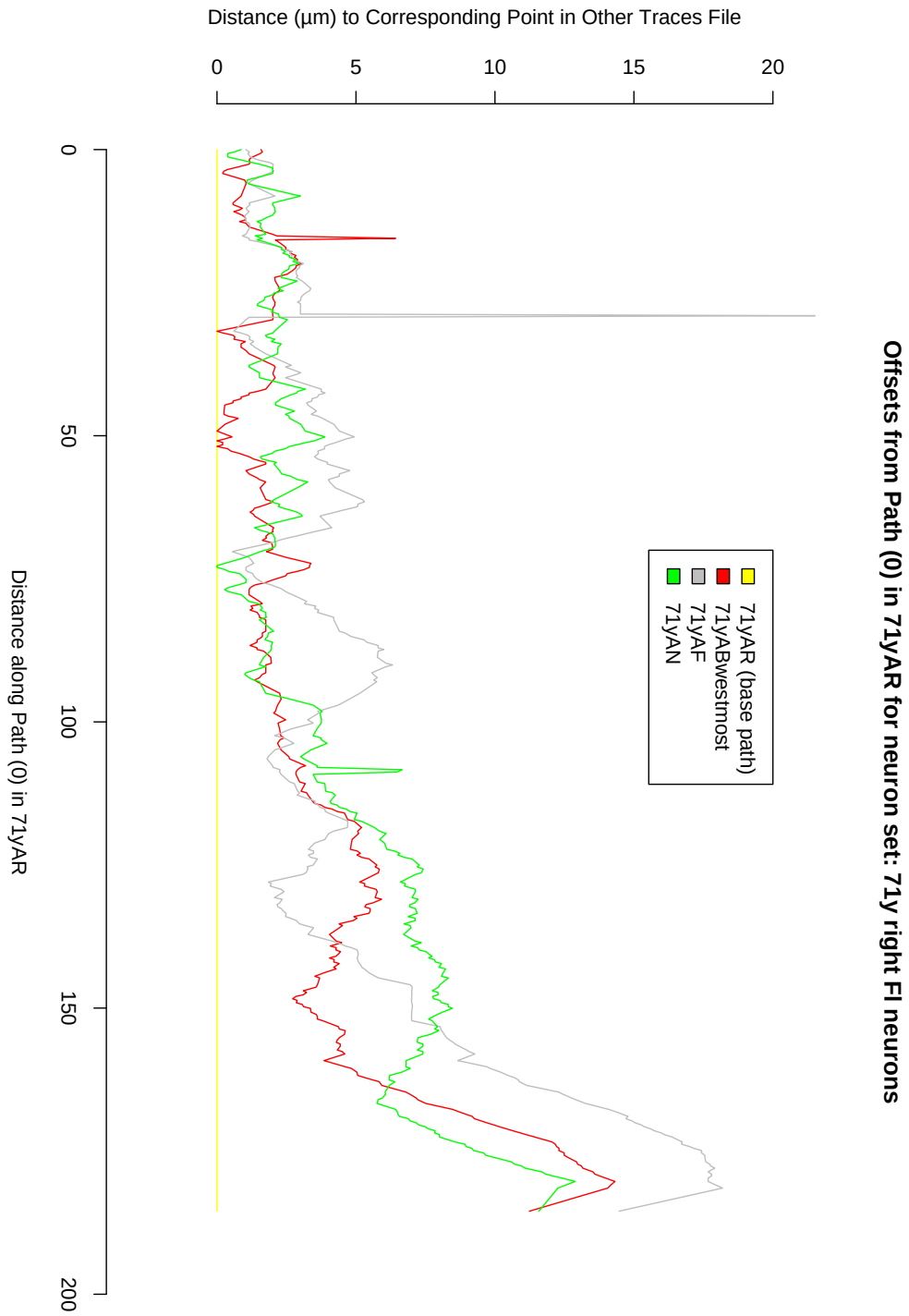


Figure 92 – A graph showing the distance to the nearest corresponding point in each other traces file to points along the typical c005 right Fl neuron. The colours of each line match those in Figure 90. The points marked in the negative y-axis region indicate where no corresponding points could be found in one of the other traces files. 0 on the x-axis corresponds to the end at the fan-shaped body, while the higher x values correspond to points at the lateral extent of the neurons.

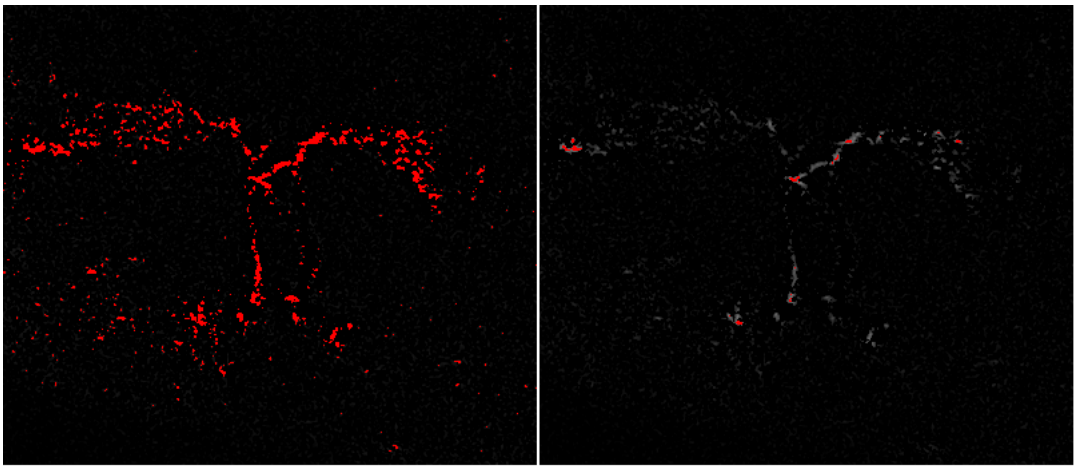


Figure 93 – The red regions in these tubeness-filtered images show the values above reasonable thresholds for r (on the left) and m (on the right).

  - Extract the most tube-like point left in P.
  - Initialize an empty hashtable H.
  - Begin a best-first search from that point using Dijkstra's algorithm and the reciprocal cost measure in the tubeness image. Carry on until a given number of iterations have been reached or a certain amount of time has elapsed, as dictated by the parameters above.
  - When each new point is discovered in the search, check whether it is either in a path found on a previous iteration or its tubeness value is above the threshold m. If so, add the point to the hashtable H.
  - Once the search has terminated we reconstruct paths from the start point to each of the points we recorded in H. There will be a lot of overlap in these paths, so if there is any part of the path that is in common with previously added paths, those sections are reused.
  - For each point in any of these paths we record the rolling tubeness value.
  - Delete any parts of the paths where the rolling average drops too low. (We want to exclude any bits of paths that might have big gaps in them.)
  - Remove from P all of the points above the tubeness threshold that we can still reach from the start after this pruning.

- Unify all of these paths to create a large symmetric connectivity graph throughout the image.

The eventual graph generated by this process is complex, so in order to reduce it to a more manageable form, I simply looked at which neuropil regions defined in a densely labelled map for that brain were connected by this graph. This was calculated by the following procedure:

- Using a thin plate spline mapping based on landmark points, I transformed a detailed label map kindly provided by Dr Arnim Jenett into the space of the traced image.
- I wrote a program which looked for every "edge link" in the connectivity graph, which I defined as connections where one point was inside a neuropil region and the other was either not in any labelled region or in a different region.
- For each neuropil region, the algorithm searched the connectivity graph for any path between any edge link of the source neuropil region to the edge link of any other neuropil region, where no other region was crossed in between. (This is an $O(n^2)$ operation, so rather slow.)
- All the shortest paths found in this way were written to a traces file in the standard format used by Simple Neurite Tracer so that they could be easily examined for false positives afterwards.

5.7 Results (Auto Tracer)

Some connectivity graphs generated with this plugin for each of the GAL4 lines under consideration are shown in Figures 95, 96, 97 and 98. These diagrams are too large, unfortunately, to be able to represent them accurately with this page size, but as indicated in the captions, full versions can be found online. Each coloured node represents a named neuropil region that some path crosses the edge of.[55] Each link between two nodes indicates that a path was found between the edges of those neuropil regions without passing through any third region. Although the printed versions are too small to see this, the edges are labelled with the names of the image files in which such a connection could be found. The more such labels appear on a given connection, the more confident we can be that this link represents a genuine common pathway in these images.

These graphs have some features that are encouraging. For example:

- The structure of the graphs have symmetry that corresponds to the bilateral symmetry of the

[55]It should be noted that since the images used in this study were cropped closely around the mushroom body, neuropil regions outside this region (e.g. the optic lobes) will not appear in these graphs.

brains. (The neuropil regions which appear on both sides of the brain are similarly coloured, so this symmetry should be apparent from the colour alone.)

- The graphs for the different lines are distinctive, and reflect to some extent features we can see in the images themselves.

However, the usefulness of such graphs is limited in a number of ways, which are described below:

- The analysis does not take into account whether there are synaptic connections at particular points on a neural pathway. In other words, a distinct and single process that passes through one neuropil region between two others with apparent arborizations will appear as three nodes, connected linearly. Since this label field completely covers the fly brain, this essentially means that there are no connections shown between non-adjacent neuropil regions.
- Even with the best available registration techniques, using a registered label field as opposed to a hand-created one for each image causes large numbers of false positives. For example, any region completely filled with tube-like structures (e.g. one of the superior layers of the fan-shaped body in 71y) is very likely to overlap with other adjacent regions. This is particularly tricky since the borders of the ellipsoid body, the fan-shaped body and the noduli are not only very close, but also indistinct due to the Z-axis blurring of confocal microscope images.

Ultimately, apparent connections generated via the Auto Tracer must still be hand-checked to find those of most interest. While my current experience with this data set suggests that the majority of these are not interesting from a neuroanatomical perspective, incorporating information about the expression pattern around each trace could help to cut out a large number of the false connections.

A further, but more minor, problem with the results from this plugin is that the paths that are found tend to be bunched together. My expectation during the development was that the paths found between distant seed points would tend to largely follow the same course, but in fact similar and parallel paths are frequently found. Performing some kind of skeletonization on the connectivity graph, to remove likely redundant paths, may produce cleaner traces.

5.8 Further Work

As it stands, Simple Neurite Tracer is a useful annotation tool for tracing neurons. Apart from a variety of user interface elements which could be improved, the area in which it would most benefit

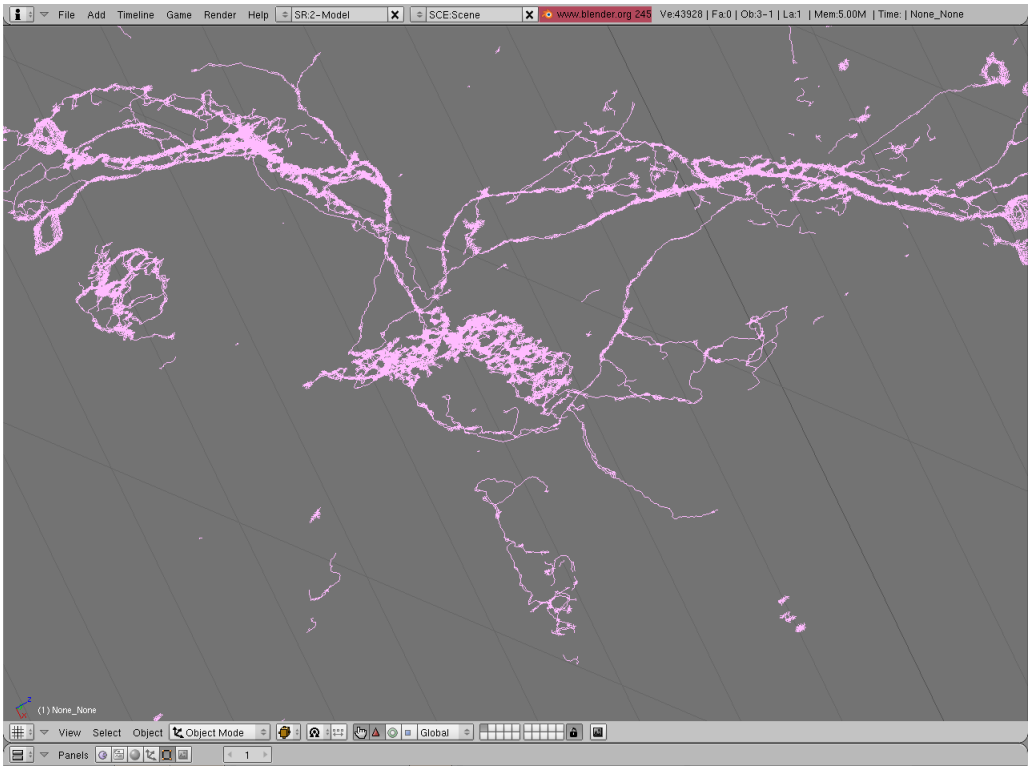


Figure 94 – Paths found by the Auto Tracer plugin rendered in Blender.

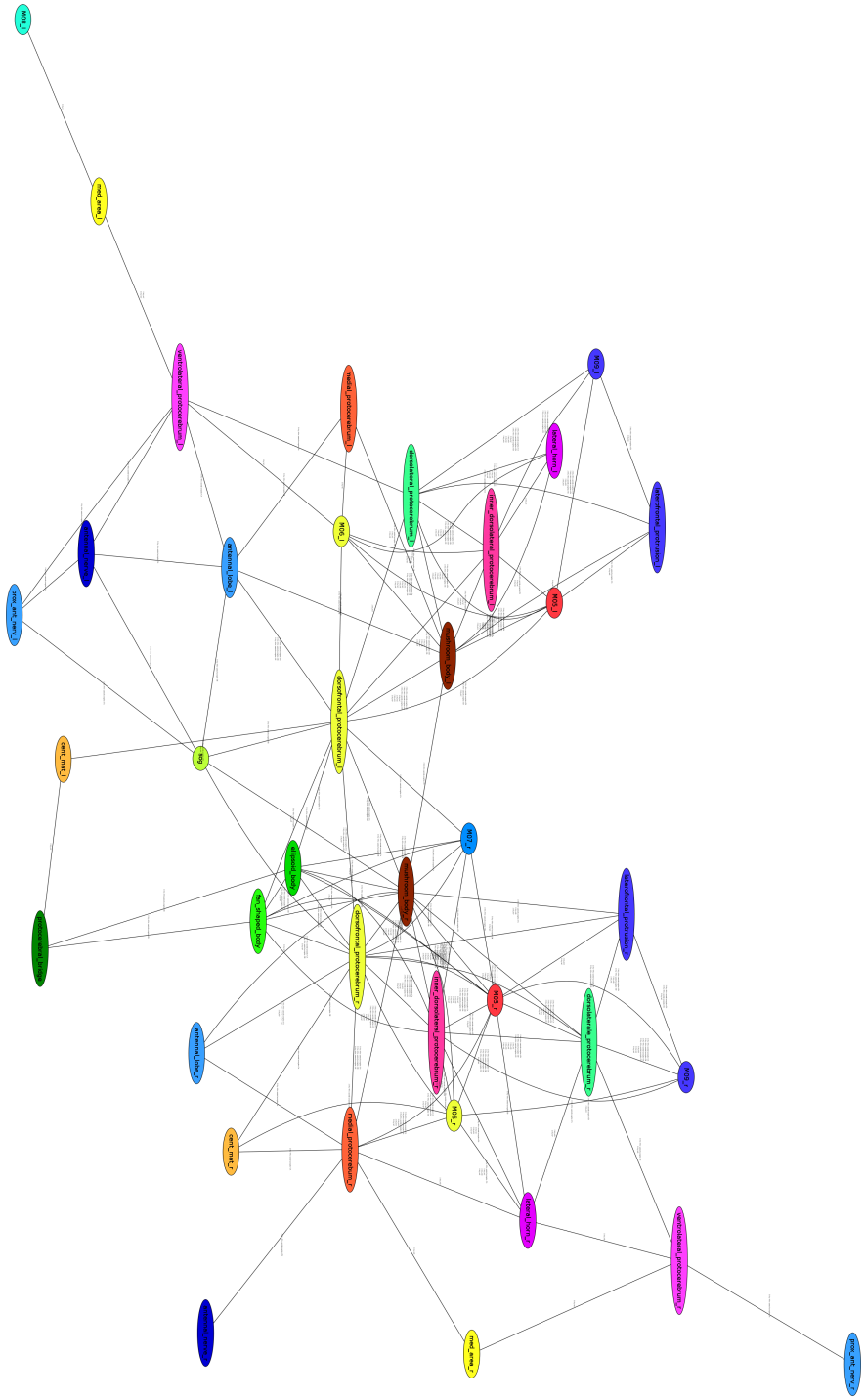


Figure 95 – A graphical representation of the results of the Auto Tracer plugin when applied to all the 210y images in the corpus. (Such graphs are too large to be easily representable in an A4 page, but full versions can be found online at `http://fruitfly.inf.ed.ac.uk/~mark/phd/`) These graphs are further discussed in section 5.7.

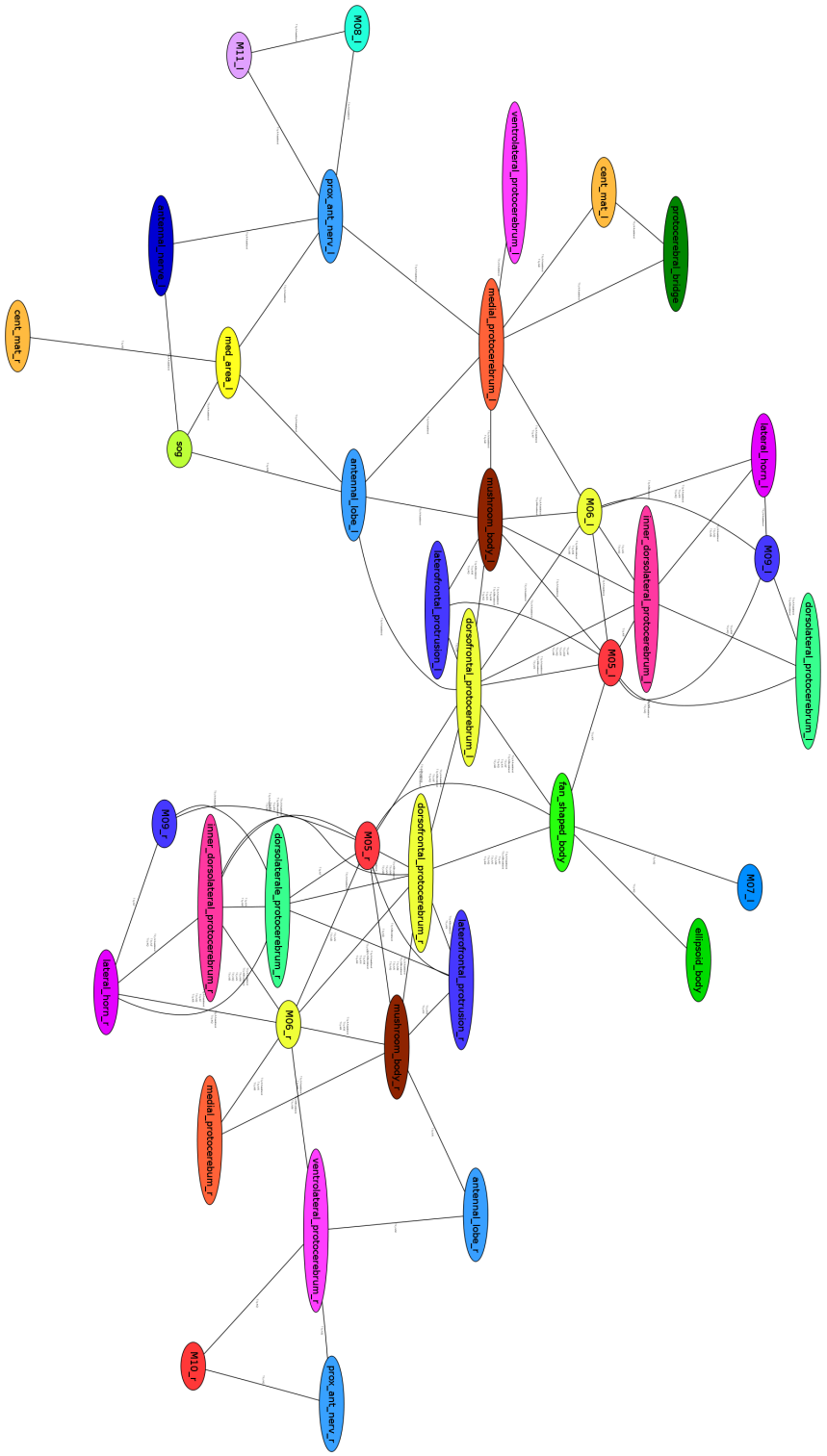


Figure 96 – A graphical representation of the results of the Auto Tracer plugin when applied to all the 71y images in the corpus. (Such graphs are too large to be easily representable in an A4 page, but full versions can be found online at `http://fruitfly.inf.ed.ac.uk/~mark/phd/`) These graphs are further discussed in section 5.7.

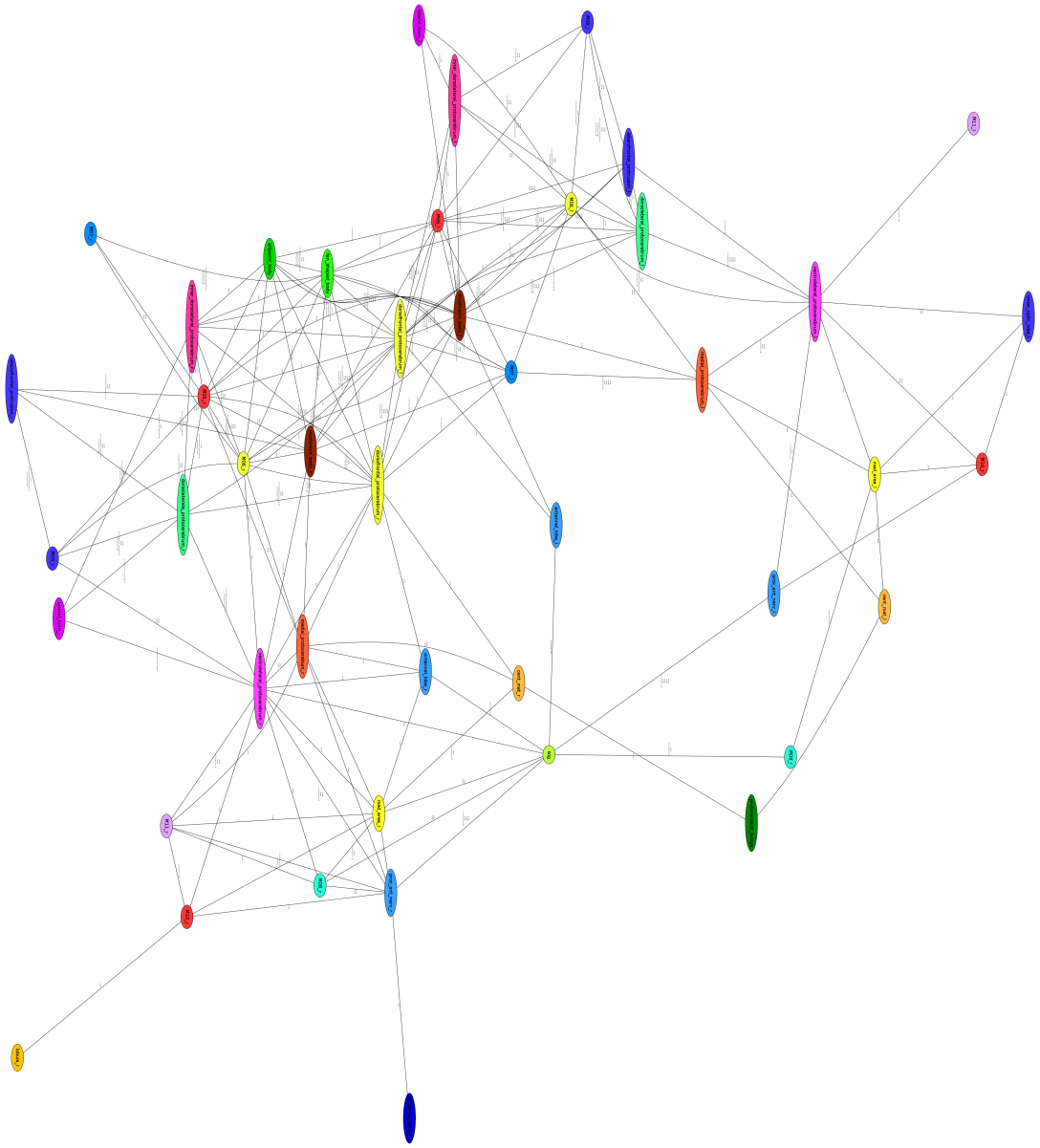


Figure 97 – A graphical representation of the results of the Auto Tracer plugin when applied to all the c005 images in the corpus. (Such graphs are too large to be easily representable in an A4 page, but full versions can be found online at `http://fruitfly.inf.ed.ac.uk/~mark/phd/`) These graphs are further discussed in section 5.7.

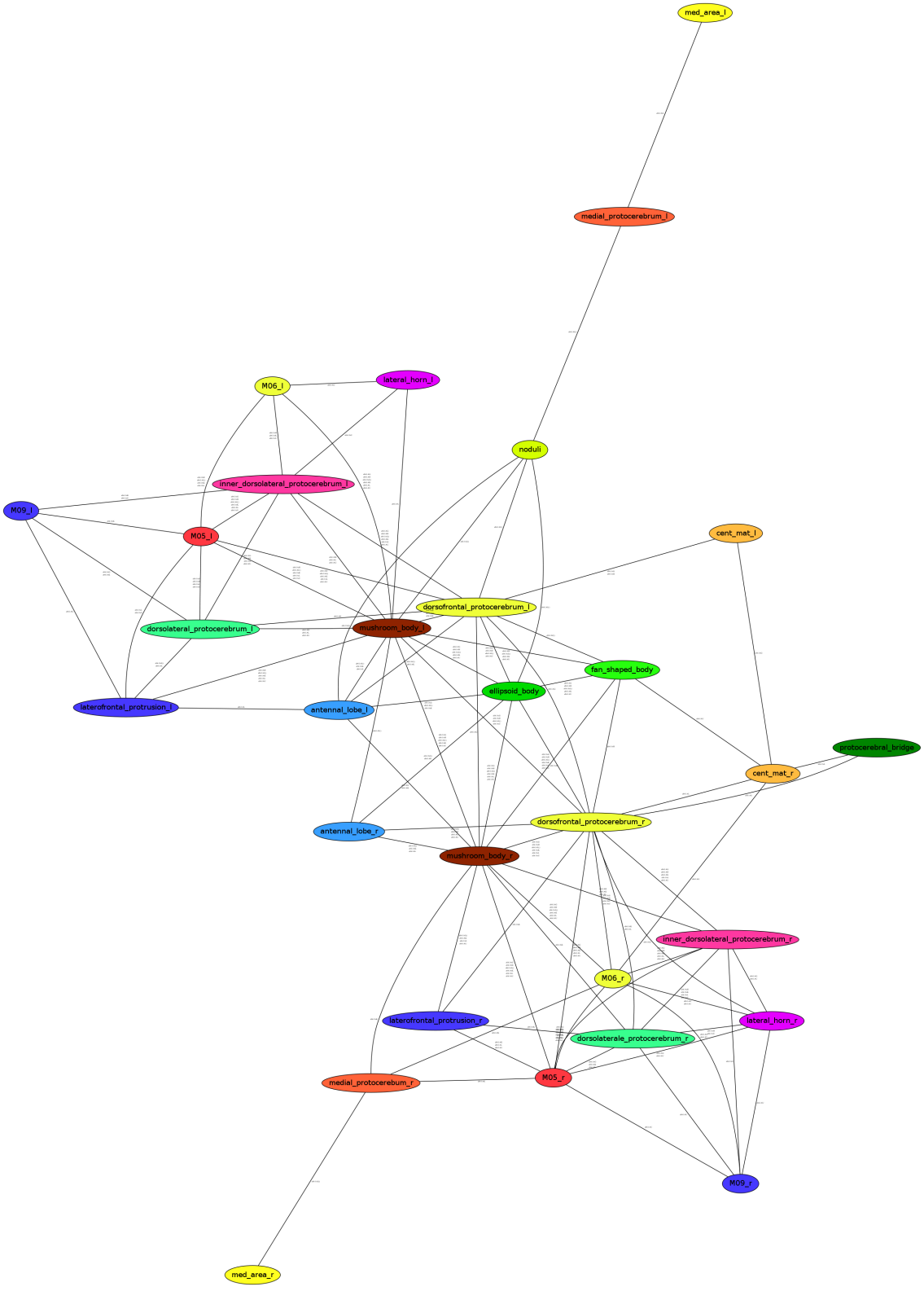


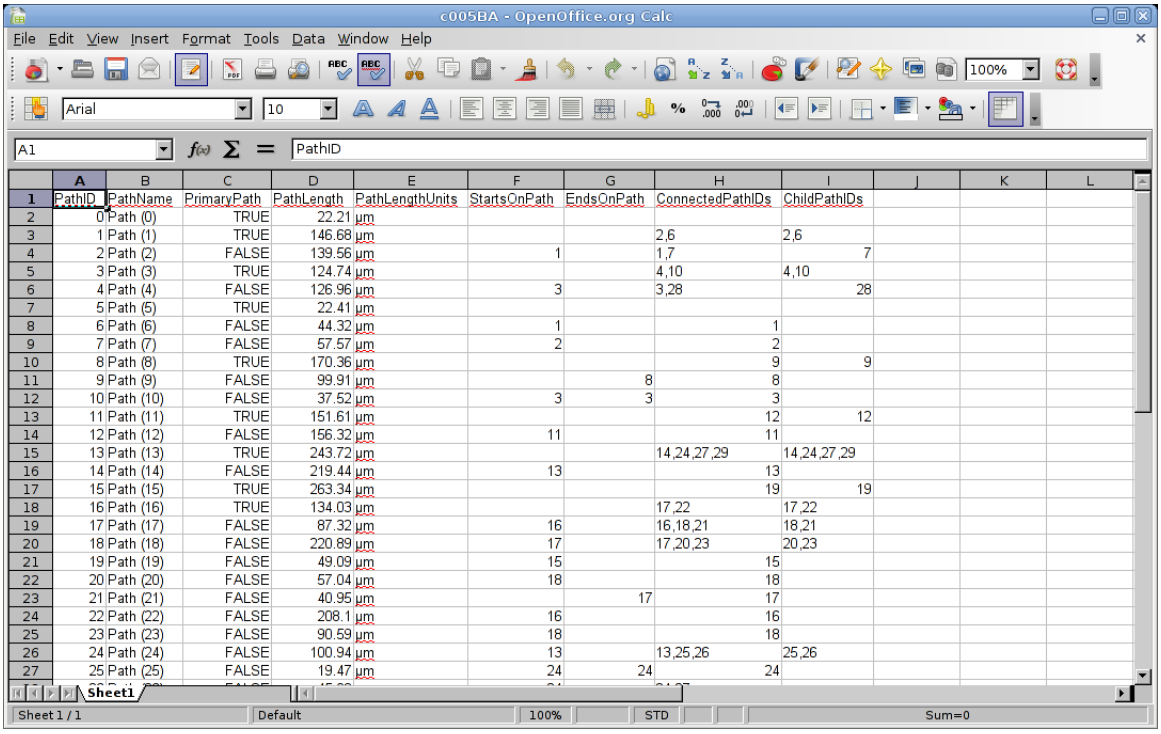
Figure 98 – A graphical representation of the results of the Auto Tracer plugin when applied to all the c061 images in the corpus. (Such graphs are too large to be easily representable in an A4 page, but full versions can be found online at `http://fruitfly.inf.ed.ac.uk/~mark/phd/`) These graphs are further discussed in section 5.7.

from further work is the reconstruction of neuronal volumes. The methods provided at the time of writing work reasonably well, but have some oddities that could be addressed. For example, neuronal volumes are easy to calculate from fills, but non-trivial to calculate from surfaces generated by the fitting method. (A user has requested that this should be made simpler.) In addition, there is currently no way provided to manually adjust the results of the "fit volume" method afterwards, which for highest quality reconstructions is essential. The quality of the automatic fitting may also be improved by switching to a more robust optimization technique, of which there are many available.

A second major area in which the Simple Neurite Tracer could be improved is in the analysis and statistics facilities that it provides. Early on in development I decided that this aspect of the plugin would remain limited, merely reporting the lengths of paths and their branching structure in the user interface. I took this decision on the basis that the features of this kind potentially required by users would be hard to predict, and it is relatively easy for someone with programming experience to write scripts that will generate the statistics they require based on the XML files saved by the plugin. However, in response to a user request, I added the option to output data as a CSV (comma-separated values) file so that the more basic statistics could be easily manipulated within a spreadsheet, as shown in Figure 99, or other statistical software. However, feedback from users suggests that they would appreciate some more analysis features built into the plugin, so at a later stage it may be worth adding such functionality.

Thirdly, the tubeness filter based on the Hessian matrix which is used in this plugin is one of the simplest alluded to in [Sato et al., 1998], and it has a tendency, due to not including the eigenvalue closest to zero, to also pick out blob-like structures of the appropriate scale. The more complex filters mentioned in that paper could easily be implemented, which may give a significant improvement to the effectiveness of this preprocessing.

As mentioned above, another area of development that could greatly benefit both the Simple Neurite Tracer and Auto Tracer plugins would be a preprocessing step to classify the type of expression found in a small region surrounding each point in an image, providing richer information about expression patterns than simply measuring tubeness. This would enable the semi-automated tracer to recognize points of paths that are in cell bodies or those which have entered regions of high expression. In the automatic tracer, it would give us a simple way of only reporting crossings into neuropil regions that are likely to be interesting. In addition, such a classifier would be very useful for producing succinct descriptions of expression patterns, of the type:



	A	B	C	D	E	F	G	H	I	J	K	L
1	PathID	PathName	PrimaryPath	PathLength	PathLengthUnits	StartsOnPath	EndsOnPath	ConnectedPathIDs	ChildPathIDs			
2	0	Path (0)	TRUE	22.21	µm							
3	1	Path (1)	TRUE	146.68	µm			2,6	2,6			
4	2	Path (2)	FALSE	139.56	µm	1		1,7	7			
5	3	Path (3)	TRUE	124.74	µm			4,10	4,10			
6	4	Path (4)	FALSE	126.96	µm	3		3,28	28			
7	5	Path (5)	TRUE	22.41	µm							
8	6	Path (6)	FALSE	44.32	µm	1		1				
9	7	Path (7)	FALSE	57.57	µm	2		2				
10	8	Path (8)	TRUE	170.36	µm			9	9			
11	9	Path (9)	FALSE	99.91	µm		8	8				
12	10	Path (10)	FALSE	37.52	µm	3	3	3				
13	11	Path (11)	TRUE	151.61	µm			12	12			
14	12	Path (12)	FALSE	156.32	µm	11		11				
15	13	Path (13)	TRUE	243.72	µm			14,24,27,29	14,24,27,29			
16	14	Path (14)	FALSE	219.44	µm	13		13				
17	15	Path (15)	TRUE	263.34	µm			19	19			
18	16	Path (16)	TRUE	134.03	µm			17,22	17,22			
19	17	Path (17)	FALSE	87.32	µm	16		16,18,21	18,21			
20	18	Path (18)	FALSE	220.89	µm	17		17,20,23	20,23			
21	19	Path (19)	FALSE	49.09	µm	15		15				
22	20	Path (20)	FALSE	57.04	µm	18		18				
23	21	Path (21)	FALSE	40.95	µm		17	17				
24	22	Path (22)	FALSE	208.1	µm	16		16				
25	23	Path (23)	FALSE	90.59	µm	18		18				
26	24	Path (24)	FALSE	100.94	µm	13		13,25,26	25,26			
27	25	Path (25)	FALSE	19.47	µm	24	24	24				

Figure 99 – A screenshot showing summary data exported from Simple Neurite Tracer viewed in a spreadsheet.

- "high expression throughout the mushroom bodies",
- "punctate expression in the antennal lobes",
- "arborizations in the inferior fan-shaped body", etc.

These summaries would be a useful addition to the online confocal image database described in Chapter 3.

The Auto Tracer plugin's most obvious problems should be addressed. I regard these as being:

1. The need to pick three parameters[56] by hand at the moment is awkward, and the choice of these parameters greatly affects the quality of the results. Developing heuristics to pick these automatically would be an important improvement.
2. Many essentially redundant paths are found in the tracing process, where several near-parallel paths are found. I mentioned one way of dealing with this above: applying a skeletonization-like postprocessing to the traced paths. However, a more satisfying solution would be to reconstruct the volume around each path after it is traced, and then ending each subsequently traced path when it meets any point within the reconstructed neuron, as opposed to meeting a point on the path.

Once these improvements are made, the automatic tracer should be evaluated against a variety of existing data sets. Fortunately, several such test sets have recently been published as part of the DIADEM challenge,[57] a competition to find software for automatic neuron tracing and reconstruction.

5.9 Summary

The tool development presented in this chapter is significant in that there is no other Free Software solution to producing this kind of analysis, from tracing to reconstruction to visualization and finally analysis. We have been able to use these tools to discard one connectivity hypothesis (a fan-shaped body to mushroom body link in c061) while at the same time generating another possibility (that such a link may exist in 210y). That the latter hypothesis seems unlikely based on other studies does

[56] To recap, these parameters are the standard deviation of the kernel used in the Gaussian smoothing, σ, and the thresholds in the tubeness image, m and r.

[57] `http://www.diademchallenge.org/`

not detract from the usefulness of these techniques. As has been often reiterated here, the results these tools can produce are most importantly limited in this study by the quality of the data they are applied to.

The combination of tracing and registration tools gives us a way to quantify variation of the paths, as shown in the graphs in section 5.5.

Finally I described some preliminary work on a largely automated version of the tracing tool. At the moment this is something of a proof-of-concept, but with the improvements suggested in section 5.8 may become a more generally useful tool.

6 Fan-Shaped Body Layers

The data that I have collected for this study of the structure of the central complex affords us an excellent opportunity to quantitatively examine the assertion in [Hanesch et al., 1989] that the fan-shaped body has 6 layers. We are interested not only in whether such a classification is supported by these data, but whether we can automatically classify layered expression in the fan-shaped body for images from arbitrary genetic lines.

6.1 Background

There appears to have been little written about the analysis of layers in the fan-shaped body in *Drosophila melanogaster*. The relevant passage from [Hanesch et al., 1989] which discusses the number of layers reads as follows:

> "The fan-shaped body consists of six roughly horizontal layers, which are slightly bent latero-ventrally. They run parallel to each other and are the consequence of stratifications of large-field neurons."

That paragraph goes on to discuss the "segment" and "shell" divisions of the fan-shaped body (orthogonal to the layers) which we will not consider in this chapter. The paper [Liu et al., 2006], which suggests a link between the fan-shaped body and visual learning, introduces the nomenclature that the most inferior layer of the fan-shaped body (marked in the lines c205, NP6510 and NP2320) is layer F1, while one of the most superior layers (marked in the lines 104y, 121y, 154y, 210y, c5, c205 and c271)[58] is referred to as layer F5. The text describing this assignment suggests that this is a "provisional" naming scheme:

> "We tentatively classify the layer stained in 104y, and the other six rescuing lines, as layer 5 (from bottom upward), and hence provisionally call the neurons F5, although, without further markers, it is difficult to reliably number the layers."

Indeed, even with markers that do express in particular layers, it is difficult to know how to count them: do drops in the expression, say, of *bruchpilot* count as layers, or only those with elevated

[58] The c5 line used in [Liu et al., 2006] is the same as that from the Fly Trap collection, so we can relate at least one of these layers to the work here.

expression? Are the layers expected to be of similar height? For example, when staining with the nc82 antibody to *bruchpilot* there are clearly three elevated layers of expression, (e.g. see Figure 53) where the middle layer is much narrower than the superior or inferior ones. We could assume that the lowest band is F1, the gap above F2, the thin middle layer F3, the gap above that F4 and the large superior band F5 - however, it is not clear that that was the intention of the authors of [Liu et al., 2006]. The papers [Hanesch et al., 1989] and [Zars et al., 2000] contain diagrams that divide the fan-shaped body into 6 layers of equal height, but otherwise I have not been able to find an alternative nomenclature suggested in the literature.

The highest estimate that I have heard of the number of layers in the fan-shaped body is contained in a discussion document prepared by Prof Kei Ito for a meeting to discuss proposed new *Drosophila* neuroanatomy nomenclature[59]. This mentions that in the fan-shaped body, "Tricolor labelling reveals up to 9 layers". The tricolor labelling in question combines cytoplasmic, pre-synaptic and post-synaptic markers.

This uncertainty about the layered structure means that this is a good problem to approach from a computational neuroanatomy perspective: perhaps we can suggest a repeatable way of classifying the layering of expression in the fan-shaped body.

6.2 Source Images

The image corpus studied in the rest of this work contains many images of four GAL4 lines with distinctive layered expression in the fan-shaped body. For this chapter I also added some images from a few protein trap lines generated by the laboratories of Dr Steve Russell and Professor Daniel St Johnston as part of their large protein trap screening project.[60] The Armstrong lab has collaborated with them to screen for YFP expression in the adult brain in each of these lines, and Seymour Knowles-Barley found several strains with layered fan-shaped body expression. I chose three of these (see Table 4) in order to supplement the data in the corpus described in section 2.7, although we discovered after collecting the data that two of the lines had the insertion in the same gene. The expression pattern for these lines was indistinguishable by eye, so I pooled the data from them. All three lines express in three distinct layers of the fan-shaped body, and since the middle one of these

[59]This document is entitled "Tentative list of the *Drosophila* neuropil regions / A draft memorandum for the nomenclature pre-meeting (10-11 May 2008 Janelia)"

[60]Cambridge Protein trap Consortium; K. Lilley, S. Russell and D. St Johnston, unpublished data

Protein Trap Line	Gene ID	Short Gene Name	Full Gene Name
CPTI-000760	CG4898	Tm1	Tropomyosin 1
CPTI-000876	CG3166	aop	anterior open
CPTI-000924	CG3166	aop	anterior open

Table 4 – Protein trap lines used in this study, courtesy of the Cambridge Protein Trap Consortium.

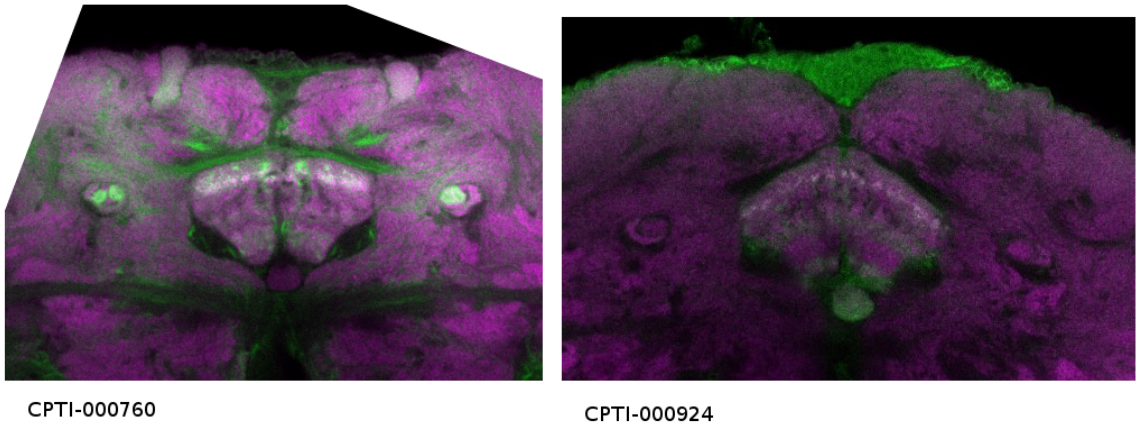


Figure 100 – Expression patterns of the protein trap lines CPTI-000760 and CPTI-000924. The green channel is anti-GFP amplified expression of YFP and the magenta channel shows the nc82 neuropil marker.

does not overlap with expression in any of the lines used earlier in this work, it seemed clear that it would be worth including these data.

As a shorthand, I will refer below to images from the line CPTI-000760 as being from "760" and those from CPTI-000876 and CPTI-000924 both as being from "924". Slices from stacks showing fan-shaped body expression of these lines are shown in Figure 100. I obtained 4 good quality images of 760 and 6 of 924, which were acquired similarly to those in the main corpus - they were co-stained with nc82 and imaged with the 40x objective, cropping the image to include at least the central complex. The immunohistochemistry used to detect the protein trap expression was similar to that used with mCD8::GFP in section 2.5.1; the insertion causes YFP to be incorporated into the protein, and the sequence of YFP is similar enough to GFP that its detection can be amplified with the anti-GFP antibody.

6.3 Methodology

This work took place before the results of Chapter 4, on image registration, had been revised and completed, so my choice of registration technique was not informed by the conclusions of that chapter. For the study in this chapter I opted to use the VIB protocol for registration, which, as described in section 4.4.4, involved individually labelling the fan-shaped body, protocerebral bridge, ellipsoid body and noduli neuropil using the Segmentation Editor plugin. The justifications for this choice of method were that:

1. The VIB protocol has been established in previous work to be effective for registration of whole fly brains.[61]

2. Since the VIB protocol works by first calculating best rigid registrations of each labelled region, the results within the fan-shaped body should be good, even if the rest of the image is misaligned.

This second point is important - although the VIB protocol performed poorly overall in the evaluation in Chapter 4, that seemed to be largely due to errors outside the fan-shaped body, which was typically aligned well.

The template brain used was the image 71yAAeastmost, which I used as a template in earlier work not described here. As can be seen from section 4.6, we would not now choose to use this brain since there are better candidates within the image corpus, but it is nonetheless one of the better images in this set.

Once the VIB protocol ran to completion, I manually went through the results and assigned by eye a score from 1 to 10 for each registration based on the apparent quality of fit of the fan-shaped body in the warped version of the nc82 channel. On this scale I discarded any images that scored less than 6, which left 38 images out of the original 57. These are shown in Table 5. The averaged image generated by taking a mean of these best registered brains has the important property that layers of high and low expression within the fan-shaped body could be easily picked out. Some of the columnar structure of the fan-shaped body, called segments in [Hanesch et al., 1989], can be made out as well.

[61] However, as far as we know, this application of the protocol to the central brain using labelled regions of the central complex is novel.

Line	Filename	fb registration score
210y	210y-40x-central-complex-CA.lsm	7
210y	210y-40x-central-complex-CB.lsm	6
210y	210y-40x-central-complex-CD.lsm	6
210y	210y-40x-central-complex-CE.lsm	7
210y	210yAC.lsm	7
210y	210yAD.lsm	8
210y	210yAE.lsm	8
210y	210yAO.lsm	9
210y	210yAP.lsm	8
71y	71yAAeastmost.lsm	10 (identical)
71y	71yABwestmost.lsm	7
71y	71yAF.lsm	6
71y	71yAM.lsm	6
71y	71yAN.lsm	8
71y	71yAQ.lsm	8
71y	71yAR.lsm	7
CPTI-000760	760-male-40x-CB.lsm	7
CPTI-000760	760-male-40x-CI.lsm	9
CPTI-000760	760-male-40x-CJ.lsm	8
CPTI-000760	760-male-40x.lsm	8
CPTI-000876	876-male-40x-CE.lsm	7
CPTI-000876	876-male-40x-CG.lsm	7
CPTI-000924	924-male-40x-CC.lsm	6
CPTI-000924	924-male-40x-CD.lsm	6
c5	c005BA.lsm	8
c5	c005BB.lsm	6
c5	c005BC.lsm	8
c5	c005BE.lsm	6
c5	c005BF.lsm	6
c61	c061AG.lsm	8
c61	c061AH.lsm	8
c61	c061AI().lsm	6
c61	c061AK.lsm	8
c61	c061AL.lsm	8
c61	c061AU.lsm	8
c5	c5xUAS-CD8GFP-40x-central-complex-BF.lsm	6
c5	c5xUAS-lacZ-40x-cc-BB.lsm	7
c5	c5xUAS-lacZ-40x-cc-BC.lsm	7

Table 5 – This table shows the images whose fan-shaped body registrations were assessed to be good enough to use in the analysis in this chapter.

I defined the points in the template space that are considered to be in the fan-shaped body to be those points that were labelled as such in at least 50% of the brains,[62] and masked out the rest of the image, leaving only the fan-shaped body region.

I considered a number of possible models for the shape of layers in the fan-shaped body. Essentially our aim is to be able to assign any point in the fan-shaped body a one-dimensional value which can be used to classify that point as being in a particular layer. Since the layers curve down laterally away from the inferior-superior axis, modelling layers as planes would not work well since each plane could cross several visible layers. Similarly, there is also curvature in each layer from the frontal to occipital sides, so the model should allow for curvature in that direction. The simplest model I could construct that might suffice for this classification was to model the layers of the fan-shaped body as being parts of the surface of concentric spheres, where the centre of the spheres is a point lying somewhere below and outside of the fan-shaped body.

Since there were three peaks in expression clearly visible in the nc82 channel of the averaged template, I began the process of fitting the centre of the concentric spheres in the model by selecting points at the peak of expression in each of the top two bands. I wrote a plugin called Fit_Sphere[63] which then optimized the coordinate of the centre of the spheres, where the "badness" of each candidate coordinate was calculated in the following way:

- For each set of points, the mean distance to the centre coordinate was considered to define the radius of that sphere. The variances of these distances for each shell were then summed, and that value was used as the measure of "badness" of the fit (i.e. the value to be minimized)

The optimization was performed with the `ConjugateDirectionSearch` class from the `pal` library. The results of the fit can be seen from the slices through the template image shown in Figure 101.

Although these results appear to be generally good, the fit of the spheres' surfaces to the layers drifts somewhat as the angle increases in the frontal or occipital direction, so I restricted the section of the fan-shaped body considered for further analysis to the points within a certain angle from the centre point. This was done by manually marking in the upper and lower bands the last points at which the fit still appeared to be good. This defined two planes that intersect in a line through the spheres'

[62] Although it may not make a noticeable difference in this case, a theoretically better method for finding the fan-shaped body region would be to use shape-based averaging [Rohlfing and Maurer, 2007].

[63] `http://pacific.mpi-cbg.de/cgi-bin/gitweb.cgi?p=VIB.git;a=blob_plain;f=vib/Fit_Sphere.java;hb=HEAD`

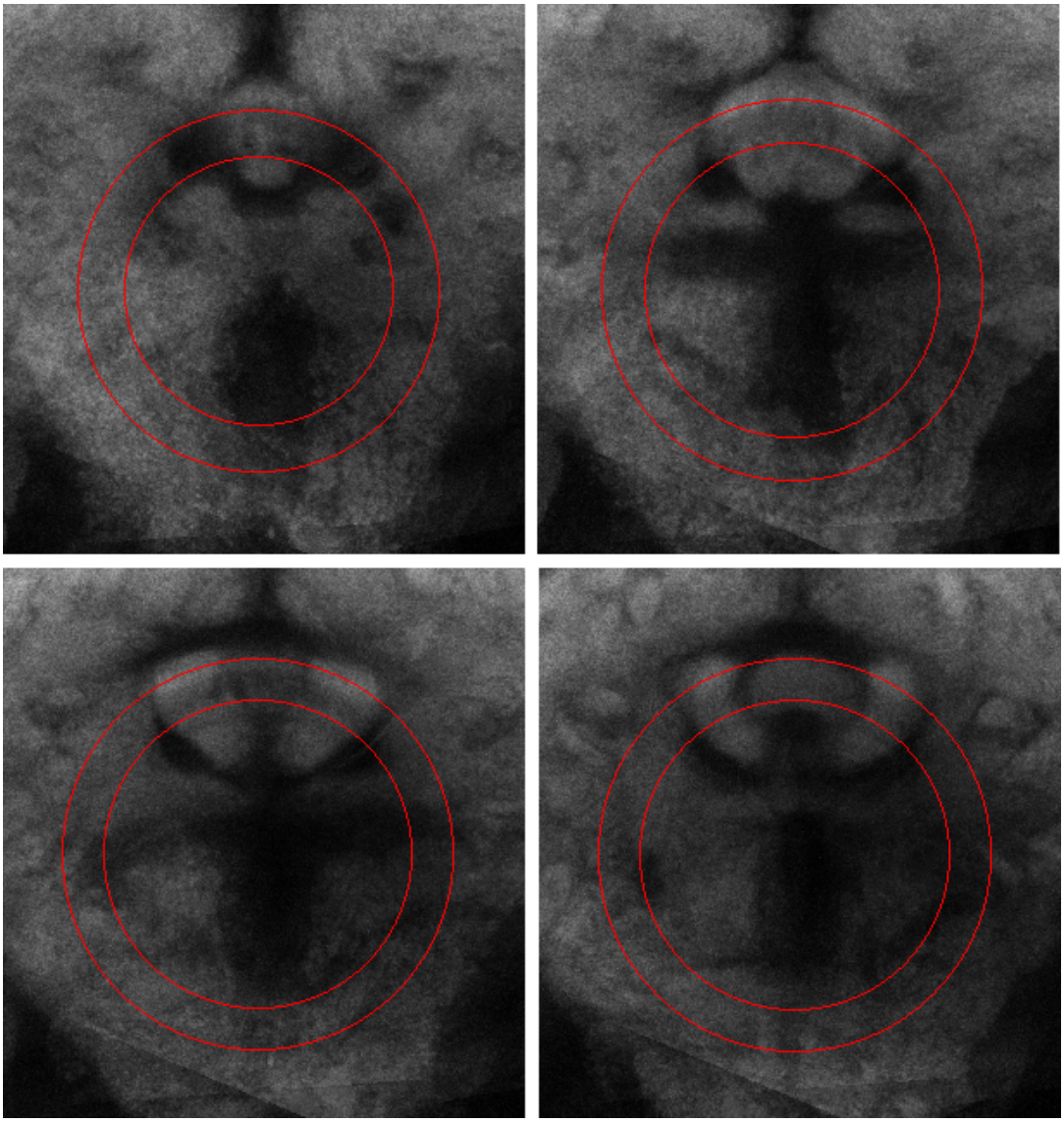


Figure 101 – Selected sections through the template with the concentric spheres fitting the top two bands of the fan-shaped body shown in red.

centre point, and only those points lying between the planes and in the fan-shaped body region were used.

6.3.1 Normalization

A problem in all analyses of this type is that imaged brains end up having very different distributions of expression, due to factors such as variation in the quality of brain preparations and the need for different PMT settings for each image. Some kind of normalization is necessary in order to compensate for these variations. For normalizing the expression within the fan-shaped body, I took the suggestion of Dr Dean Baker to use a technique from microarray analysis known as "quantile-based normalization". This is one of the techniques described in [Bolstad et al., 2003]. The plugin that I wrote to do this normalization is documented and available on the web[64] but I will briefly describe the algorithm here.

Any number of images, image stacks or regions of images may be used as input to the algorithm, but to simplify the terminology of this explanation, we will assume that the input is a number of images. For each input image all of the values contained in that image are ranked, and then divided into a number of quantiles. For each quantile we then calculate the mean of all the values in that quantile across all the images. Finally, we replace all of the values in each quantile with that mean. This has the effect of providing robust histogram equalization across any number of images. An example of the effect of this equalization is shown in Figure 102.

6.4 Results

In the discussion that follows "height" in the fan-shaped body is used to mean "distance from the centre point of the fitted fan-shaped body spheres".

For each image stack I binned the normalized values into 100 shells of about 0.47 micrometers thickness, so that each data point in the later graphs is derived from hundreds of values. The graphs obtained by plotting the mean value in each of these bins against the height of the bin are shown in Figures 103 to 108. I have not plotted standard error for the values in each bin since these are very large, both due to noise in the acquired images and the punctate nature of the expression in some

[64] `http://homepages.inf.ed.ac.uk/s9808248/imagej/quantile-normalization/`

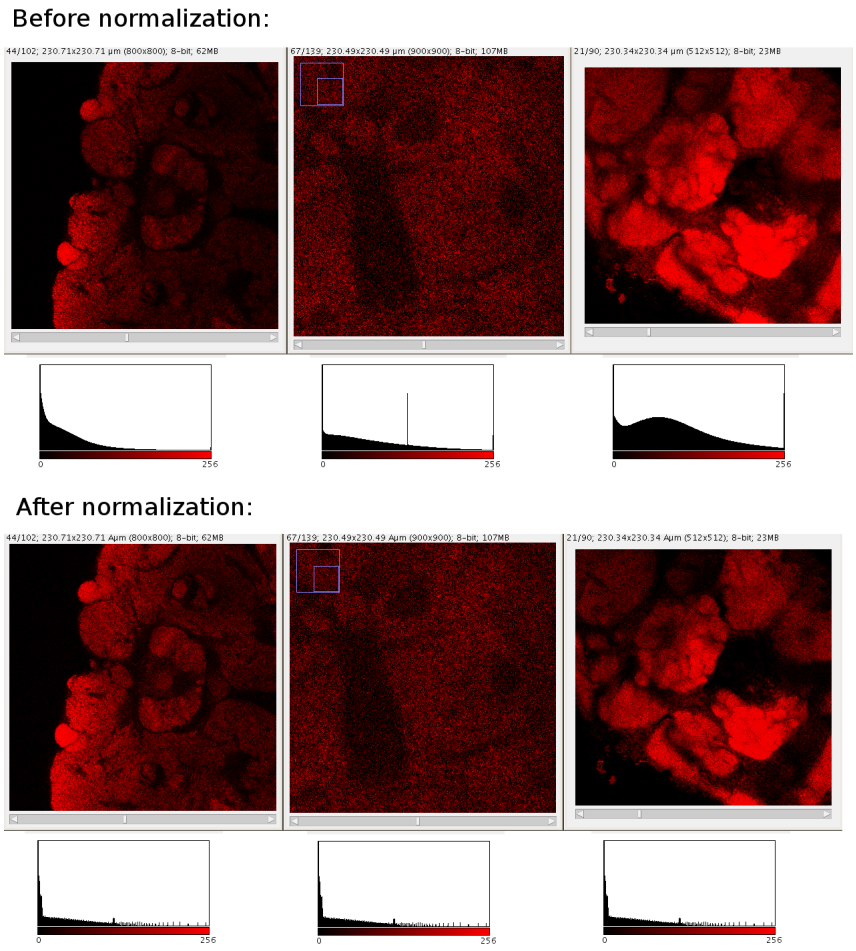


Figure 102 – An image showing the effect of applying quantile-based normalization to three images. At the top are shown the three original images, with corresponding histograms showing the distribution of values within those images. The three images below show them after the normalization step - it should be clear that the image on the left has been brightened and that on the right has been slightly dimmed. The corresponding histograms of values have been equalized.

layers and lines; for example, the peak expression in 71y is clearly divided into separate processes, so most of the values are either close to 0 or 255. The curves based on these means are smooth, however, and do correspond closely to the variation in expression that we can see by eye, as the representative slices shown underneath each graph should confirm.

These graphs show one particularly striking feature: the shapes of the curves for each image are very similar, but they are significantly offset from each other. Although this might be due to the expression pattern varying in different brains, some inspection of the warped nc82 channels of the source images suggests that this is in fact due to misalignments arising from the registration algorithm. One contribution to this error is that the rigid registration used in the VIB protocol does not adjust for different scales of brains, so if the fan-shaped body is larger in one brain than another, that will generate some error even if after rescaling they could match up perfectly. In addition, there will be some error arising from imperfect registration. Unfortunately, this error tends to be large compared to the thinner layers that we might hypothesize to be distinguishable. For example, in c005 the variation in offset of the peaks is of the same order as the depth of the layer of elevated expression.[65]

Figure 109 shows similar expression curves but with the data for every image of each line pooled. In other words, each point represents the mean of values in a 0.47 micrometer deep shell across all the images of that line. As discussed above the offset of the data means that the precise shape of these curves should be treated with caution. However, it does provide a useful overview of the broad differences in the expression through the fan-shaped body in the different lines. It is clear that there is no simple layer classification obvious from inspection of these graphs.

A different view of the expression in each line is shown in Figure 110. This shows the distribution of the maxima of the expression curves for each line, with each point being the mean position of the local maxima and the error bars representing 95% confidence intervals around those means. Since there are clearly bimodal (210y) and trimodal distributions (nc82, 924 and 760) in certain lines, appropriate numbers of local maxima are found in those cases. The intention of this presentation of the data is that it should give us a way to visualize the offset of the curves.[66] If it appeared, for example, that the peaks in expression of 71y and 924 were non-overlapping despite the offset (and,

[65] Another way of looking at these offsets, however, is that the sharper peaks (e.g. in the c005 graph) give us an upper bound on the contribution of all sources of error in this reduced dimension view of the data, a useful statistic.

[66] There will be also be some variation due to certain peaks being flatter than others, but with the sharper curves this will be dwarfed by the offset error. As an alternative, I tried fitting a Gaussian to each peak and using the mean instead of the maximum value, but the fitting failed in large number of cases.

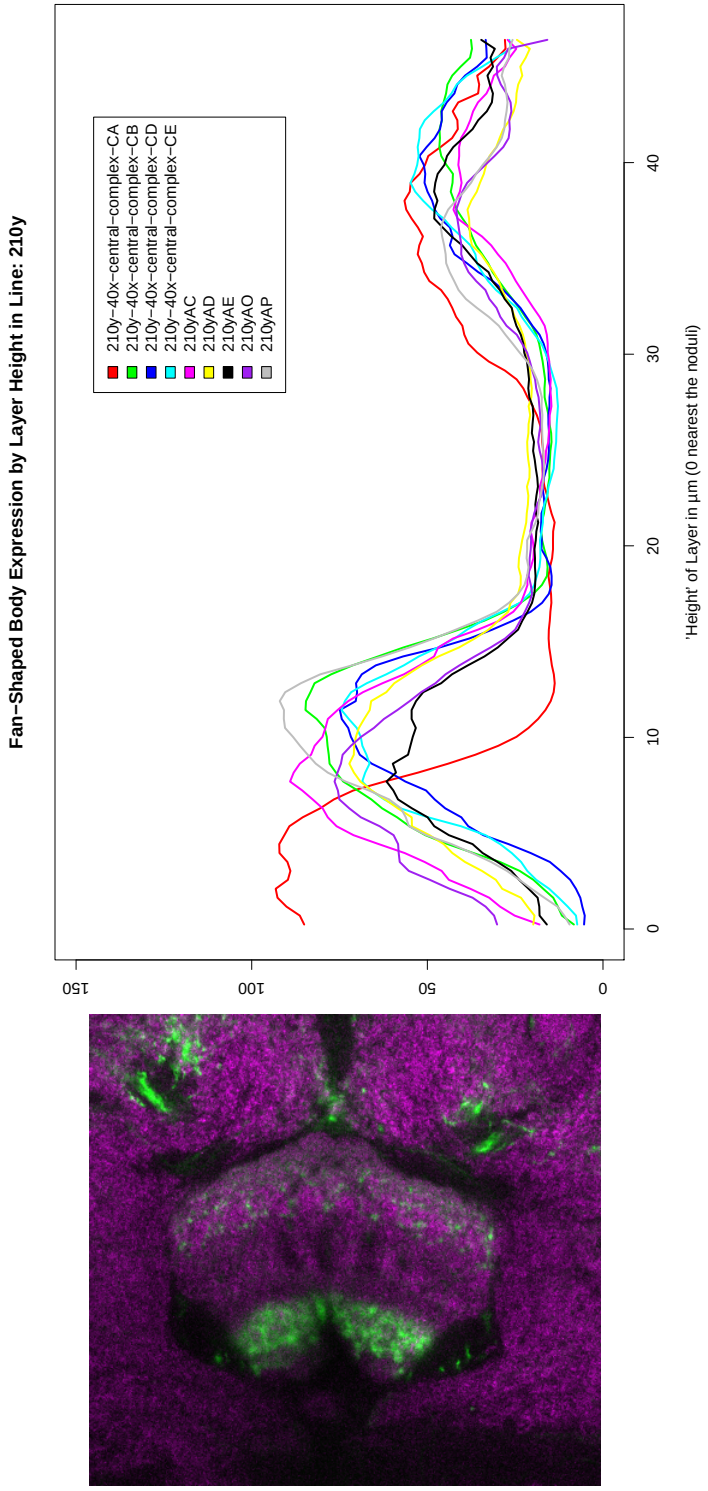


Figure 103 – Expression level of the line 210y by height in the fan-shaped body for several image stacks. Each data point represents the mean value of voxels in a band 0.47 micrometers high at a particular height in the fan-shaped body.

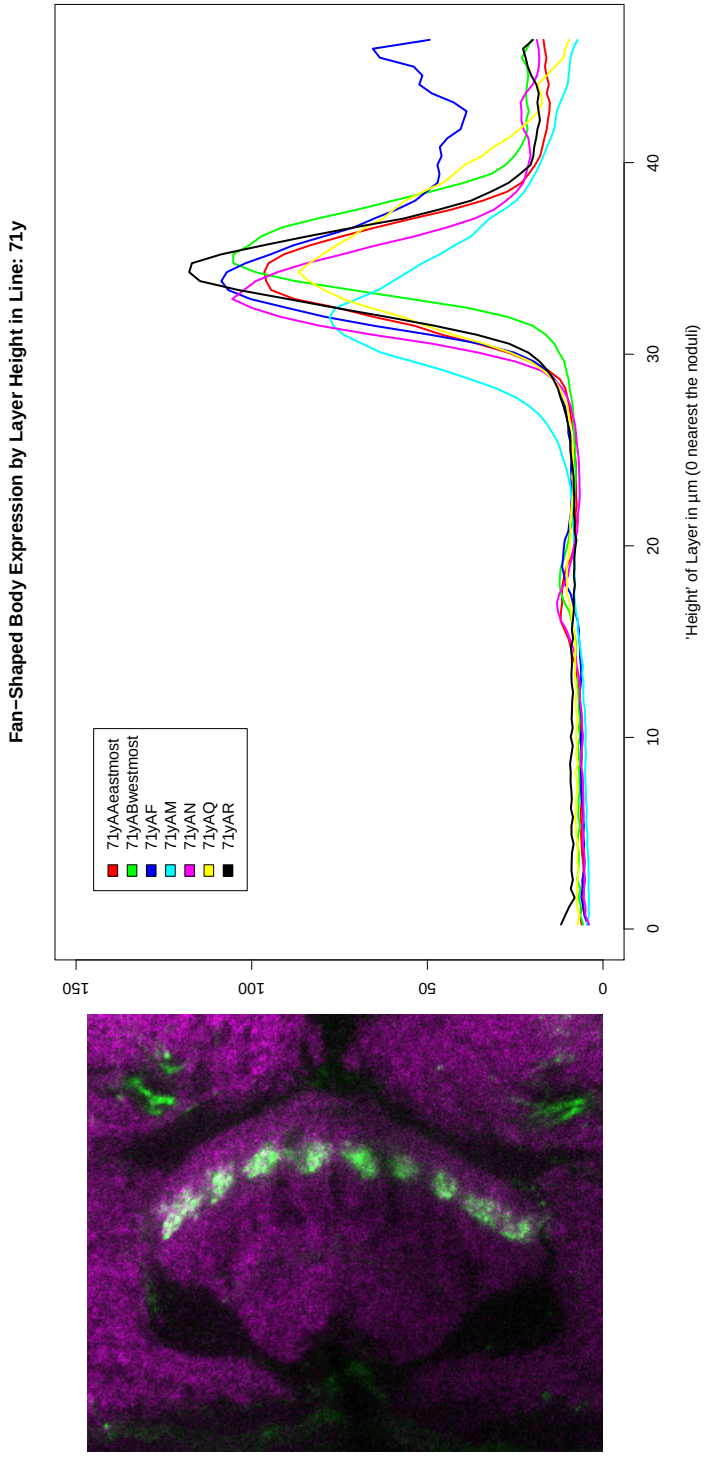


Figure 104 – Expression level of the line 71y by height in the fan-shaped body for several image stacks. Each data point represents the mean value of voxels in a band 0.47 micrometers high at a particular height in the fan-shaped body.

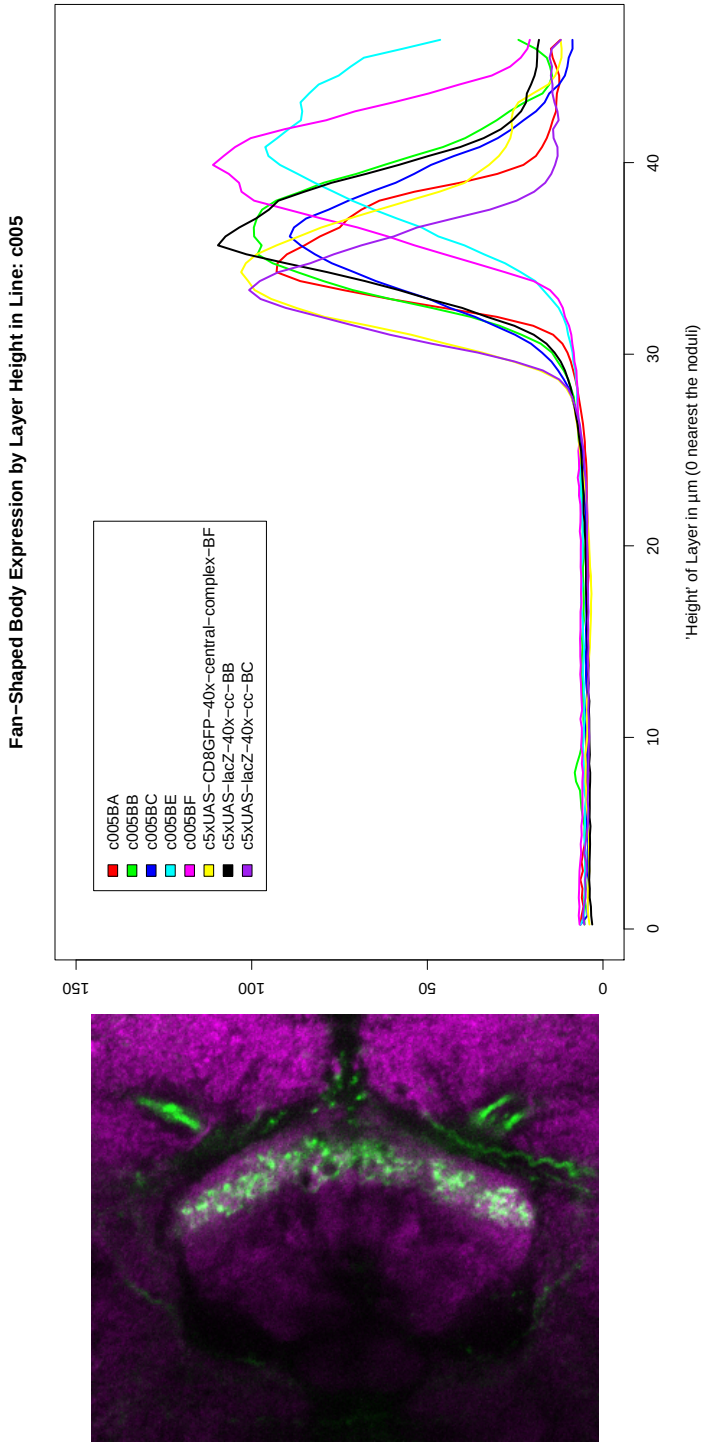


Figure 105 – Expression level of the line c005 by height in the fan-shaped body for several image stacks. Each data point represents the mean value of voxels in a band 0.47 micrometers high at a particular height in the fan-shaped body.

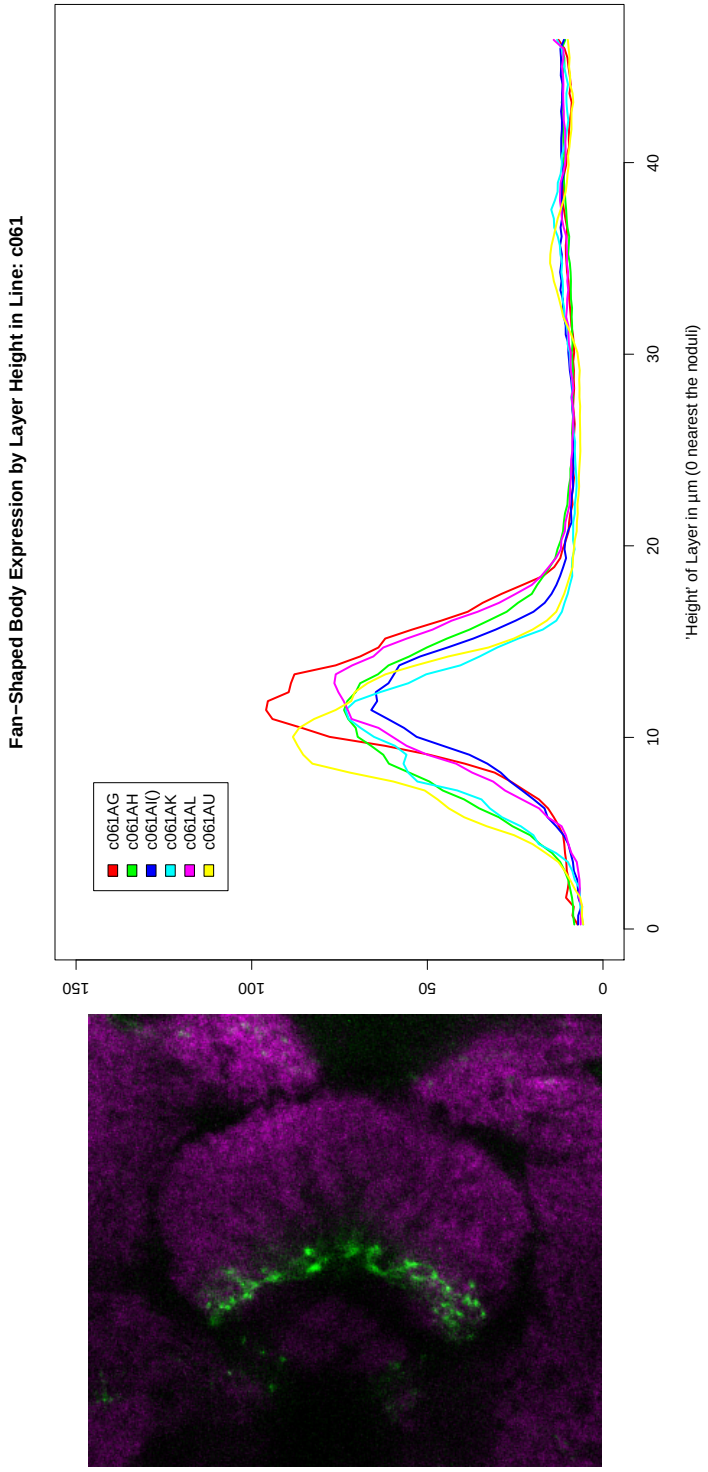


Figure 106 – Expression level of the line c061 by height in the fan-shaped body for several image stacks. Each data point represents the mean value of voxels in a band 0.47 micrometers high at a particular height in the fan-shaped body.

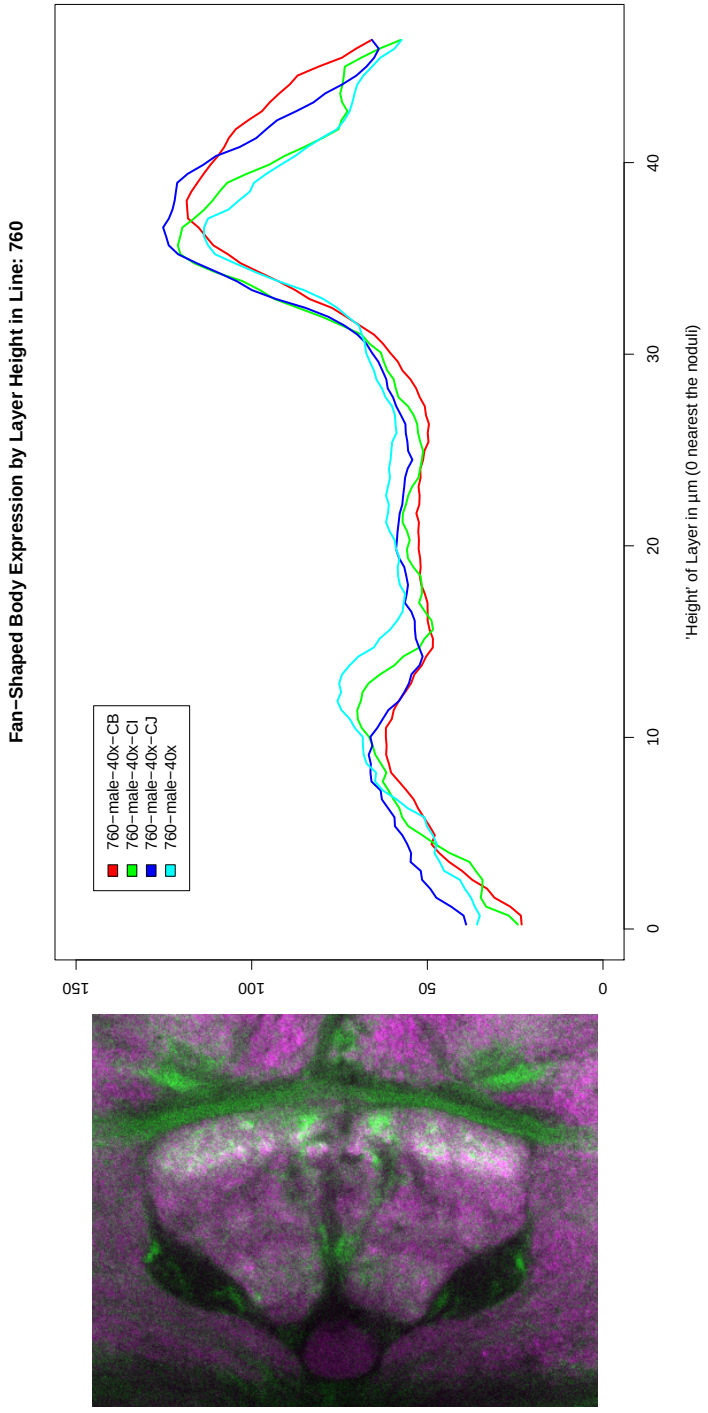


Figure 107 – Expression level of the line 760 by height in the fan-shaped body for several image stacks. Each data point represents the mean value of voxels in a band 0.47 micrometers high at a particular height in the fan-shaped body.

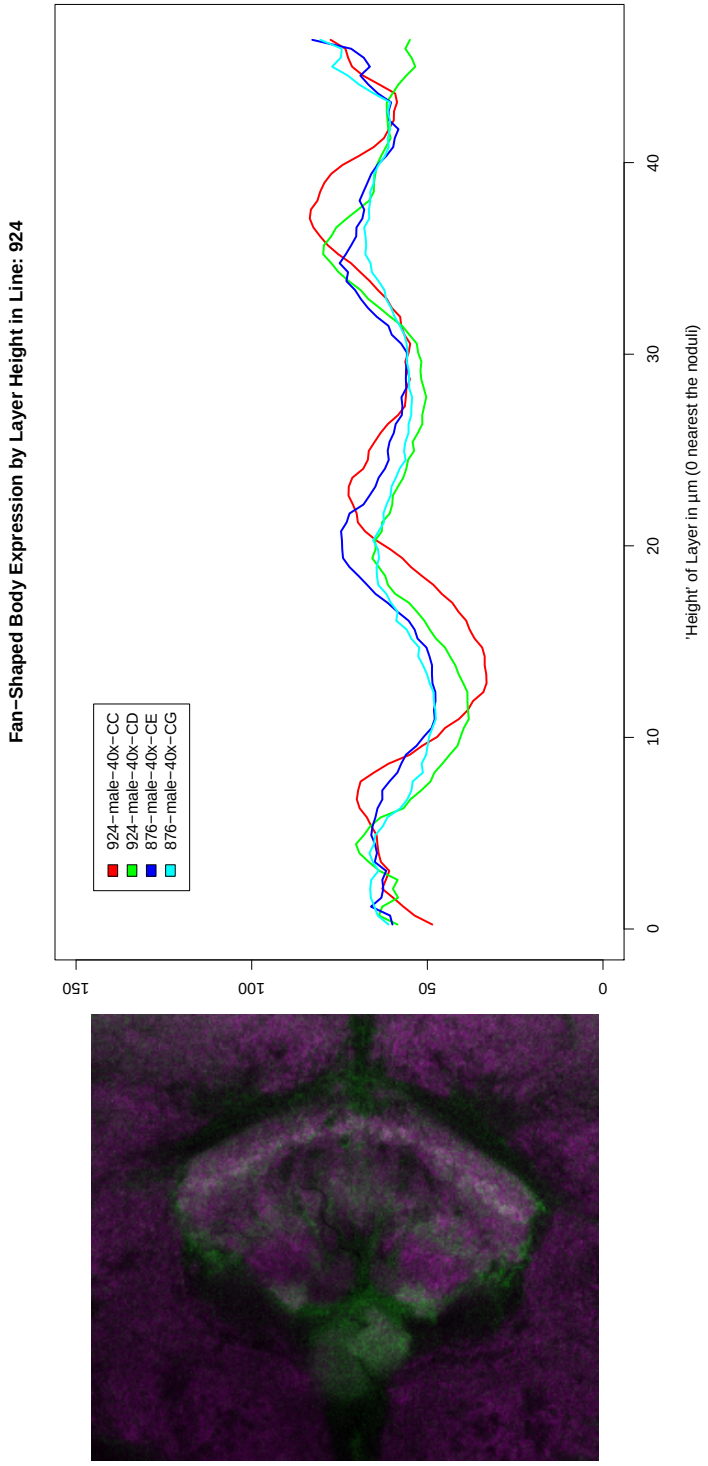


Figure 108 – Expression level of the line 924 by height in the fan-shaped body for several image stacks. Each data point represents the mean value of voxels in a band 0.47 micrometers high at a particular height in the fan-shaped body.

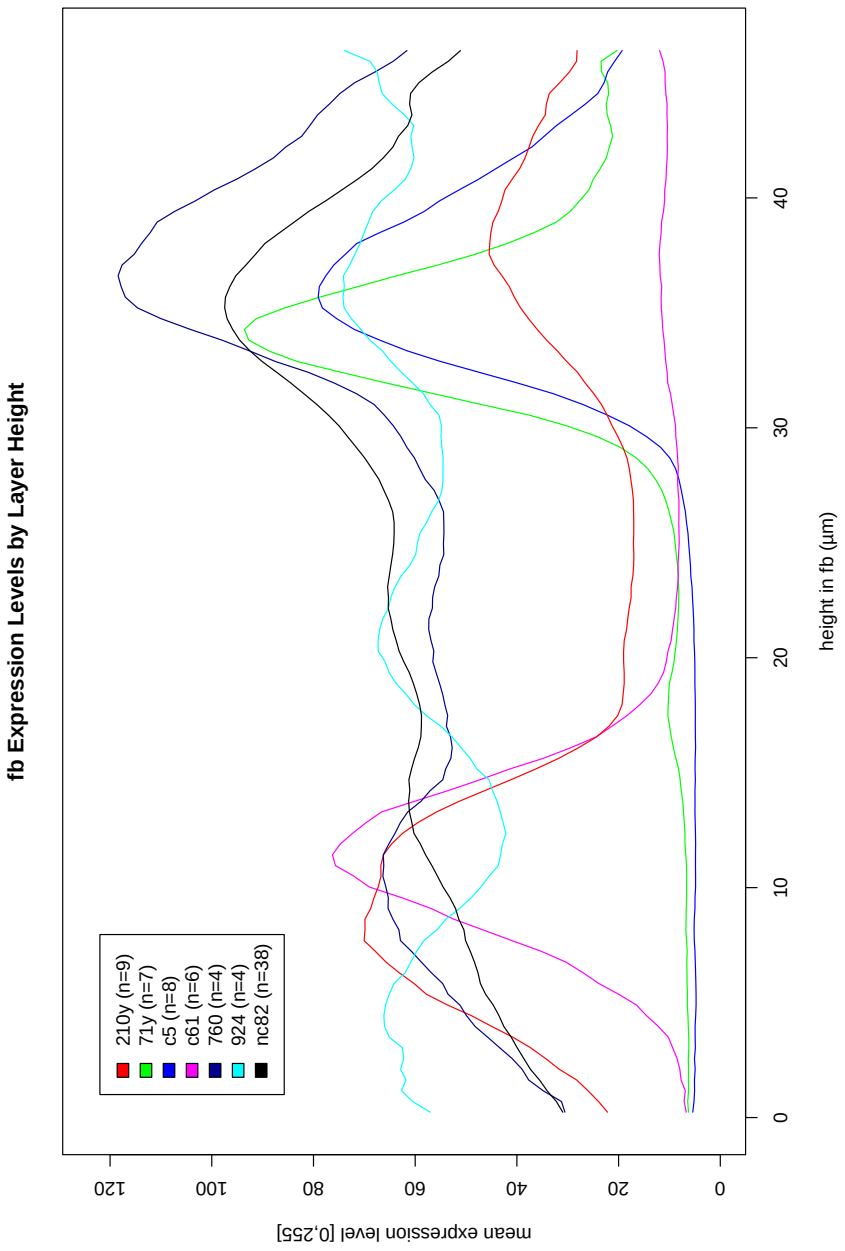


Figure 109 – Each of these curves, which shows level of expression by height in the fan-shaped body, is made up by pooling data from each of the lines. (i.e. each curve is made up of an aggregation of the data in each of the graphs in Figures 103 to 108.)

furthermore, we demonstrated that this was significant via hypothesis testing) we would be able to use this as a basis to suggest that these markers pick out distinct layers in the fan-shaped body. However, it is clear from Figure 110 that this is not the case.

Ideally, we would like to be able to infer from these data what the "best" model of layers in the fan-shaped body is according to an appropriate complexity criterion. For example, one suggestion is to assume that each of the source layers are well-modelled by a Gaussian distribution about some mean height in the fan-shaped body and that all the curves produced by any genetic line are made up of weighted sums of these Gaussians. However, aside from appealing to the central limit theorem and the appearance of the distributions in particular lines, we have no reason to expect that the base distribution of each layer is of that shape. An approach which requires fewer assumptions is to follow the method of [Jefferis et al., 2007] and use Independent Components Analysis (ICA) to decompose the data into a number of different source signals. The classic example of a situation in which ICA might be used is that of "the cocktail party problem", where one attempts to separate into source signals (e.g. individual voices) the recordings from several microphones placed in a noisy party [Hyvärinen and Oja, 2000]. The analogy here is that the underlying layers of neurons are like the voices at a party and mixing them with various weights should produce the graphs above. In order to experiment with this, I used the fastICA algorithm [Hyvärinen, 1999] in GNU R with between 1 and 10 sources to see whether the extracted source signals might represent distinct layer-like peaks. It became apparent, however, that the results from ICA were biologically implausible since the source signals and their coefficients had negative components, and the fluorescence in different layers can only be additive. An alternative algorithm which similarly splits the input data into source signals but which enforces a non-negativity constraint is "Non-Negative Matrix Factorization" (NMF) [Lee and Seung, 1999, 2000]. I translated Patrik O. Hoyer's simple Matlab implementation[67] of this technique with a Euclidean cost function into GNU R, and tried it with between 1 and 20 sources. The graphs in Figures 112 and 113 show the sources factored out when these different numbers of sources are specified, and Figure 111 shows the root mean squared error in the expression curves reconstructed from these sets of sources.

It is pleasing that this technique does reduce the expression curves into plausible underlying models of individual layers. In general, I think this is a promising technique for extracting such models from data. However, there are a couple of different reasons why we cannot conclude from these data what

[67] `http://www.cs.helsinki.fi/u/phoyer/software.html`, accompanying [Hoyer, 2004]

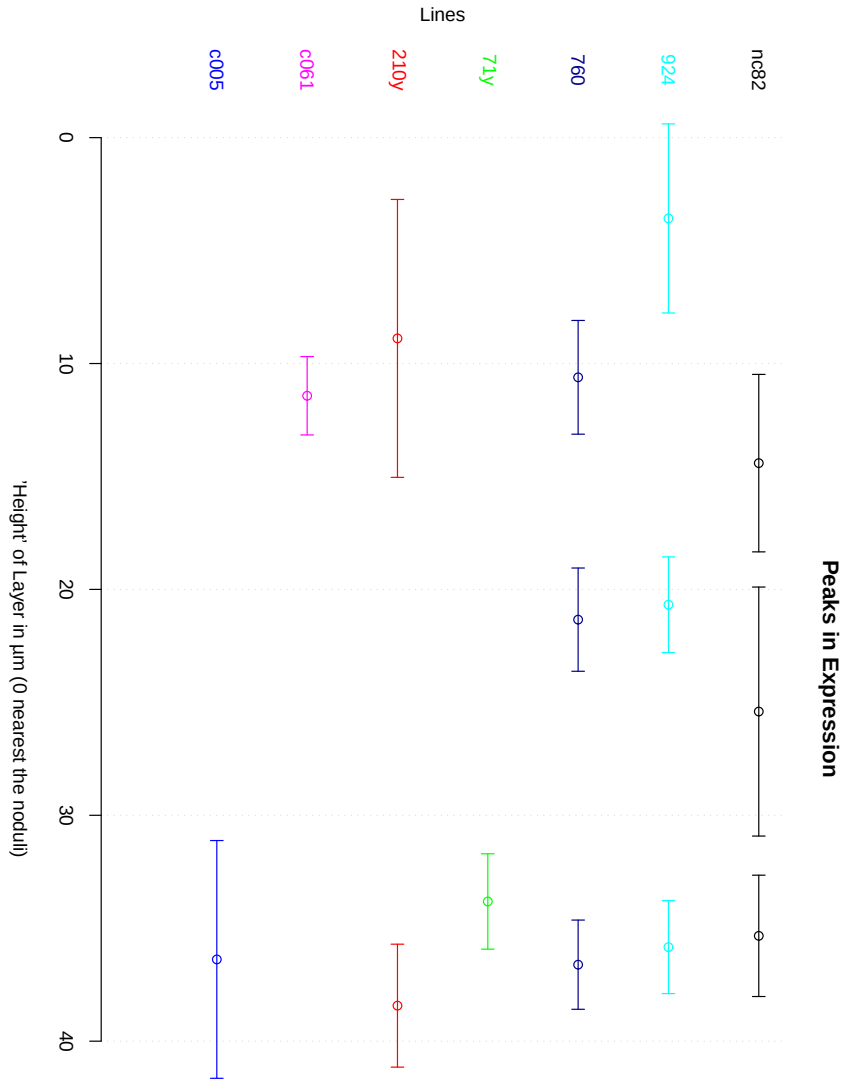


Figure 110 – These graphs show the mean peaks of expression with a 95% confidence interval. I manually picked between 1 and 3 ranges to look for peaks in, since some are clearly bimodal (210y) or trimodal (nc82, 924 and 760) curves.

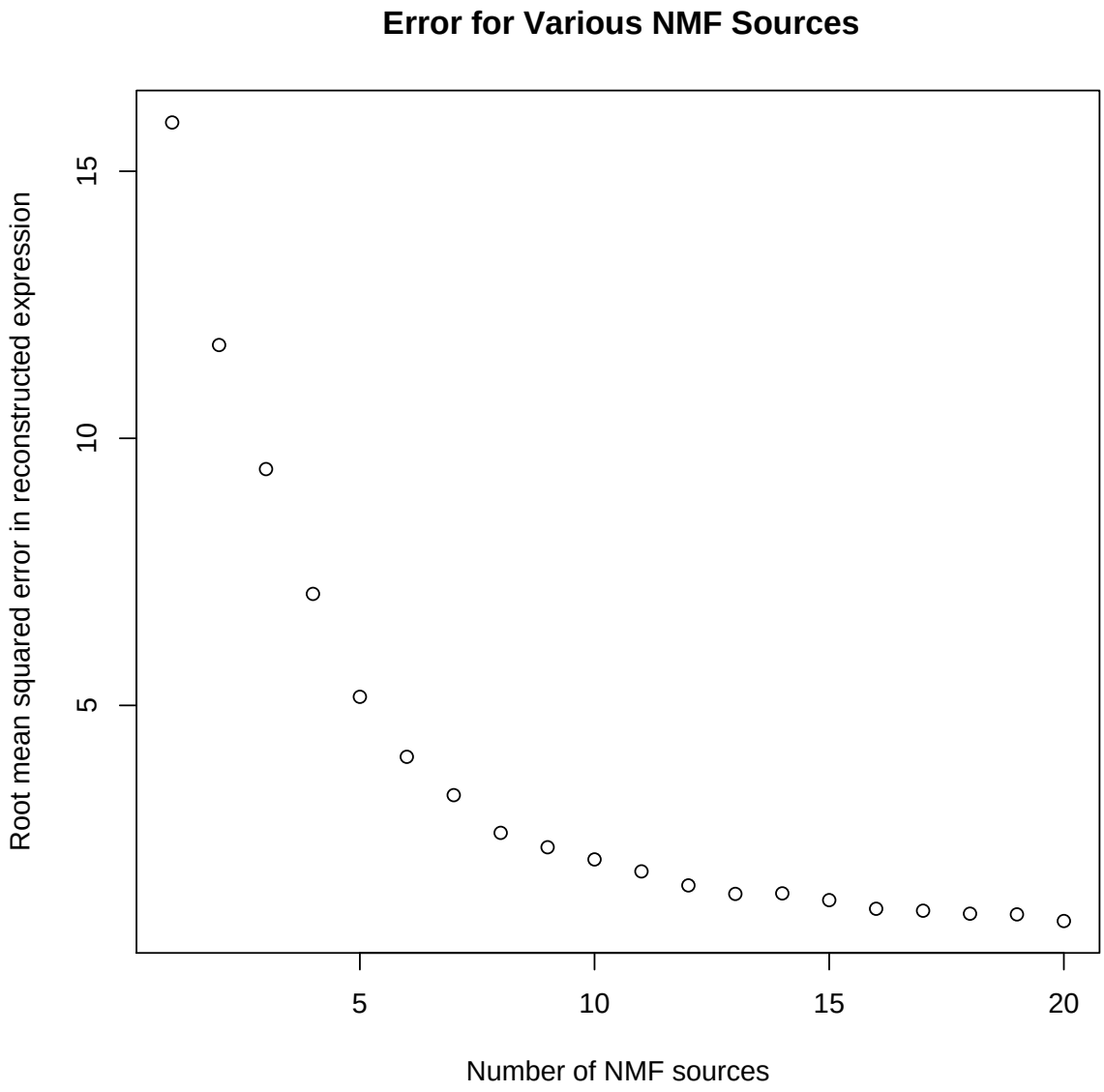


Figure 111 – The root mean squared error in expression curves reconstructed from NMF-extracted sources for different numbers of sources.

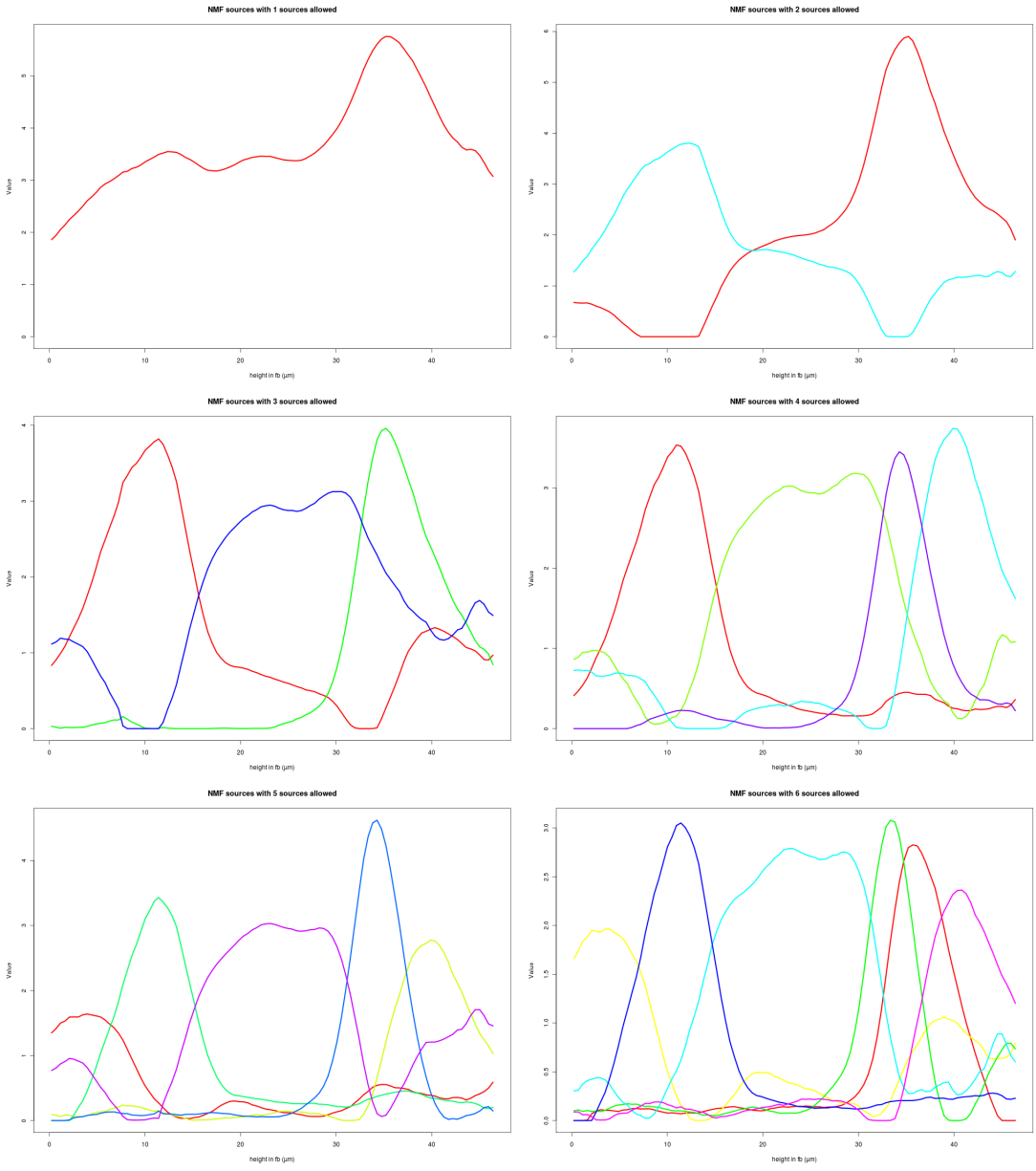


Figure 112 – Source signals discovered by NMF for when between 1 and 6 source signals are specified. (Larger versions versions of these graphs can be found at `http://fruitfly.inf.ed.ac.uk/~mark/phd/` although the overall shapes of the graphs are much more relevant than the (arbitrary) units on the y-axes. As with the previous graphs in this chapter, the x-axes represent height in the fb.)

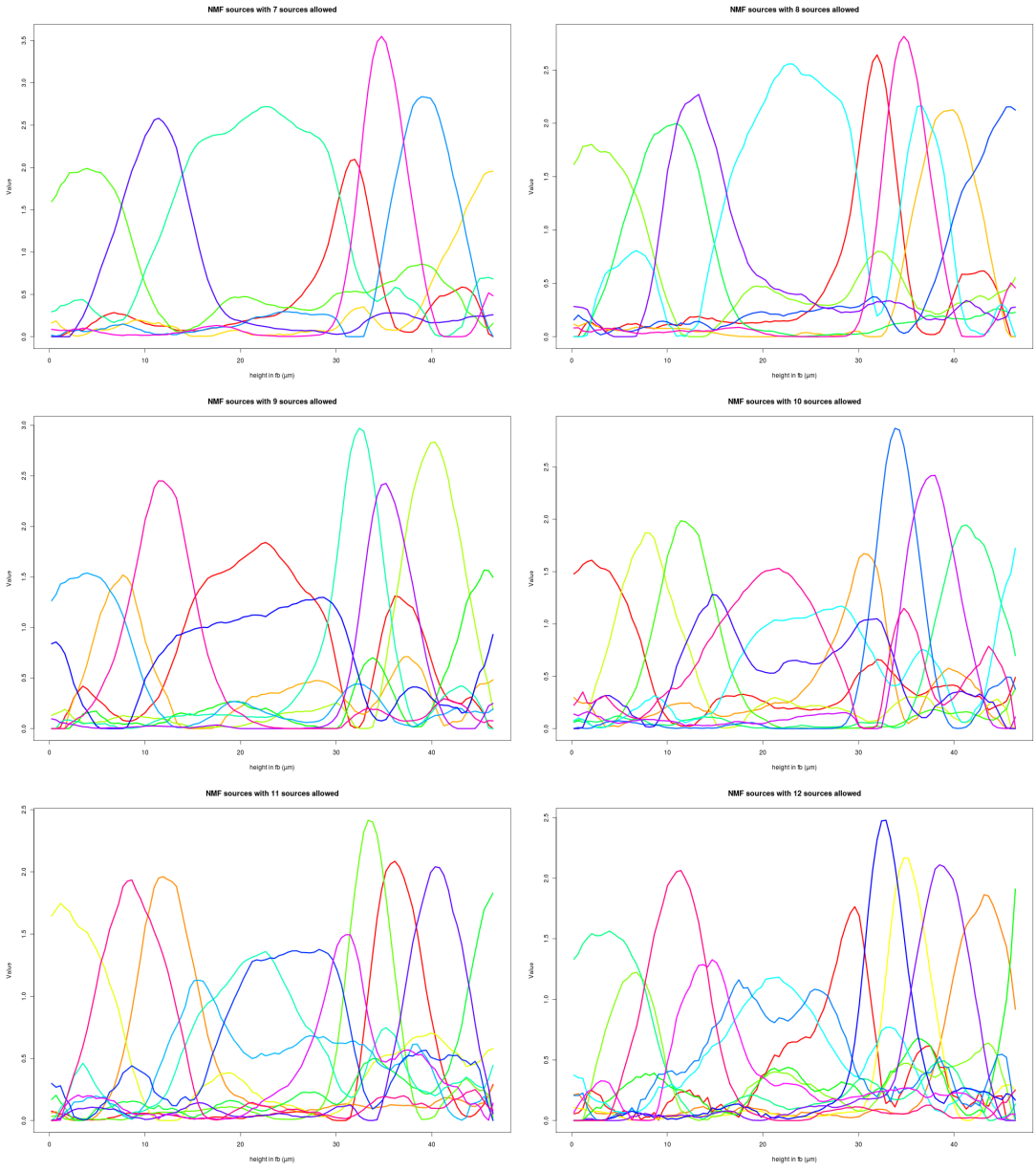


Figure 113 – Source signals discovered by NMF for when between 7 and 12 source signals are specified. (Larger versions versions of these graphs can be found at `http://fruitfly.inf.ed.ac.uk/~mark/phd/` although the overall shapes of the graphs are much more relevant than the (arbitrary) units on the y-axes. As with the previous graphs in this chapter, the x-axes represent height in the fb.)

the best underlying layered model for the fan-shaped body is:

1. The results from the NMF are not unique. The algorithm I have used is guaranteed to always improve its fit on each iteration, but starts from random data, so is not necessarily improving towards a global minimum. In addition, there are variants of the algorithm that optimize for a particular "sparseness" of the sources, which would in this case correspond to curves with less overlap.

2. Even with as few as 5 sources the error in the reconstructed curves is similar to the error between curves from the same line, suggesting that with numbers of sources in the region of previously hypothesized numbers of layers (i.e. 6 to 9), this is likely to be overfitting the source curves to errors in the image registration.

Nonetheless, this family of algorithms represents a very promising approach to this data, and with improved data and registration results, the error and sparseness criteria may provide a good way to evaluate the different models.

6.5 Summary and Further Work

The method described above to reduce expression data in the fan-shaped body to values at particular heights is a useful approach to simplifying the analysis of expression within this neuropil region. This can be seen from the sharp peaks in the graphs for the lines with sparser expression, such as 71y (Figure 104) and c005 (Figure 105). To summarize briefly, the method I would apply in order to classify the expression pattern of a new line would be as follows[68]:

1. (Take the template brain, manually mark points at the peak band of expression in several layers in the nc82 channel.)

2. (Find via numerical optimization the centre point which best fits the points in those layers to spherical shells.)

3. Register the new candidate image to the template image based on the nc82 channel.

[68]The items in parentheses are only necessary once for each template - typically one will only pick a new template infrequently.

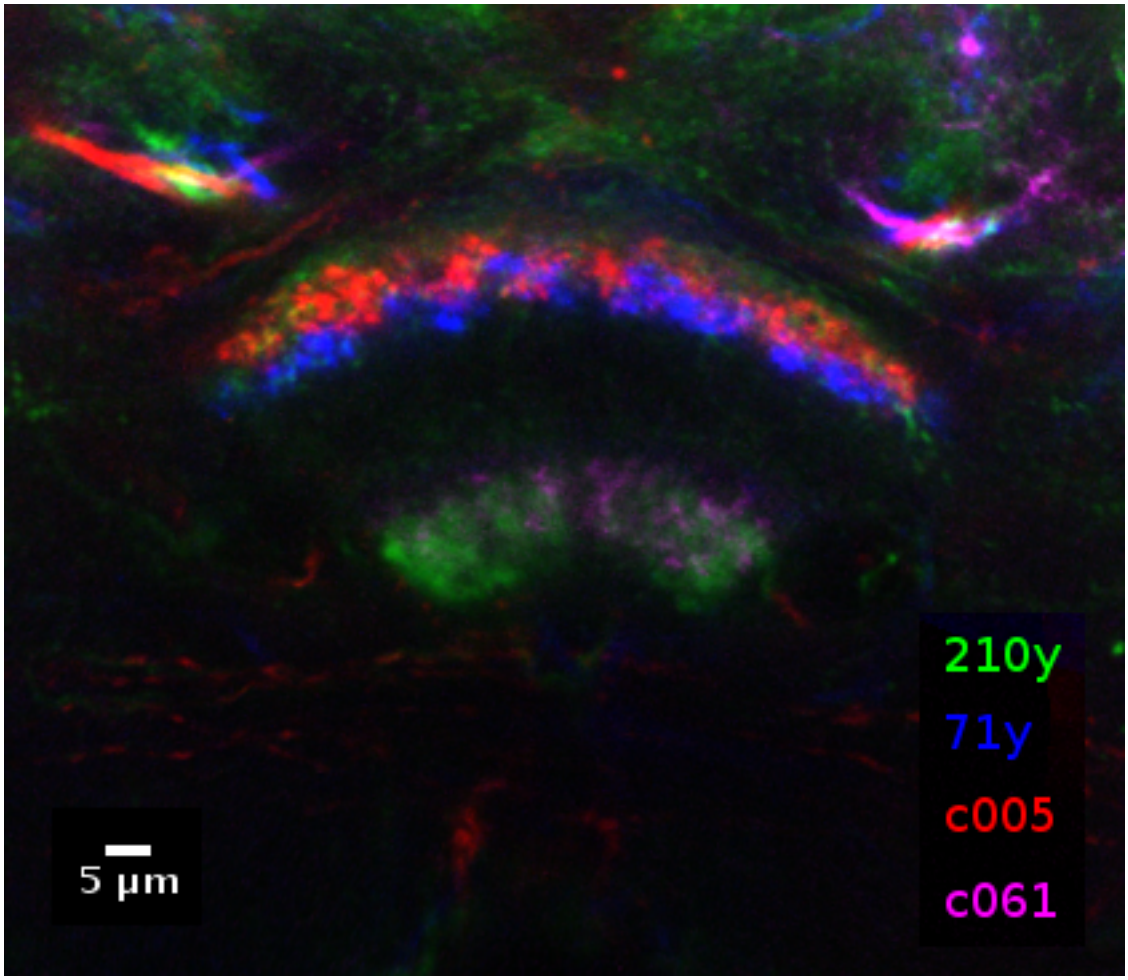
4. Calculate the level of expression of the reporter channel between narrow spherical shells centred on the point found in step 2.

If time allowed, the next step that I would take in order to get clearer results from this analysis is to switch to using CMTK for the registration. This method has the advantage that it will compensate for different sizes of fan-shaped body in different brains, and was demonstrated in Chapter 4 to produce better results in general than the VIB protocol.

In order to have good enough data for this method to allow us to propose a new model of fan-shaped body layers, both the source data and image registration need to be of very high quality; this is clear when we look at the very best registered images to see whether the different lines appear to be expressing at different heights. In these images we see some support for the idea that there are layers of the fan-shaped body which are finer grained than any 5 or 6 layer models I have heard proposed. For example, consider the top image in Figure 114: this shows the reporter channel of the four best-registered images from each of c005, c061, 210y and 71y, where the errors in registration of the nc82 channels are very low. (The bottom image in Figure 114 shows how well the nc82 channels of the c005 and 71y images are aligned.) The offset in the upper image strongly suggests that c005 and 71y are picking out different classes of neurons which could define thin layers within the fan-shaped body. Indeed, a similar separation is weakly suggested by Figure 110.

It may be that the differences between identifiable layers in the fan-shaped body are so fine that aggregating data with image registration is counter-productive. The remark of Prof Kei Ito that up to 9 layers are distinguishable in tricolor labellings of the fan-shaped body perhaps suggests that it may be more productive to focus on the analysis of such images individually.

However, if we were to carry on with the methodology presented here to produce a more complete dissection of the fan-shaped body's layered structure, the most useful next step would be to collect high-quality data from new lines and markers which show different layered expression in the fan-shaped body. Whereas switching to other registration techniques is a relatively quick task, finding new lines to add to the analysis is potentially very time-consuming. There are many labs around the world with libraries of genetic lines that may be useful for this purpose, and typically it is necessary to personally visit the labs in question in order to view images derived from these constructs.



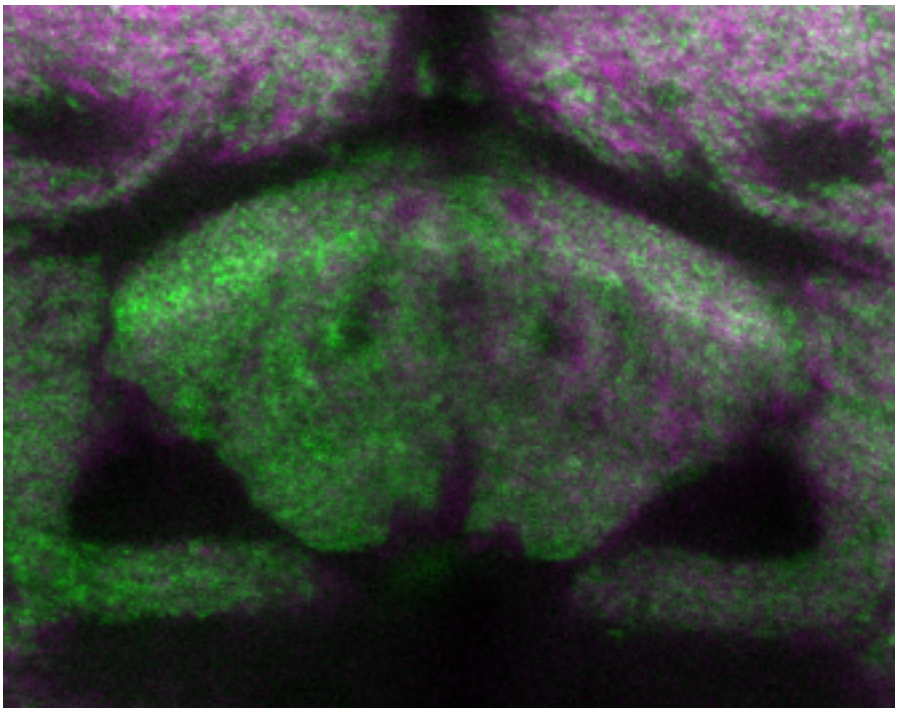


Figure 114 – Top: an example image showing overlays of the four best registered images from c005, c061, 210y and 71y. Bottom: the corresponding nc82 channels for the c005 and 71y images used for the top image, showing that the registration error is much less than the apparent offset in the neurons that are picked out in those lines.

7 Discussion

This project began with an ambitious goal: to build a computational atlas of connectivity to the fan-shaped body, and then use those data for modelling neural circuits. However, having had this eventual aim in mind has informed and directed the work towards finding pragmatic methods for using confocal microscope data for this purpose. This discussion chapter is split into section 7.1, which discusses possibilities for future work, and section 7.2, which summarises the work and conclusions of this thesis.

7.1 Further Work

Near the end of each preceding chapter was a brief discussion of possible further work which was closely coupled to the content before it. In this chapter, however, I discuss either more general concerns or distinct projects which were provoked by the earlier work.

The first section deals with the key issue of acquiring more useful image data (section 7.1.1). Then I discuss some preliminary work on inferring neuronal polarity using preferentially distributed markers and acquiring high contrast images from multiple scans (section 7.1.2).

7.1.1 Data Quality Issues

In this project, the single greatest obstacle to building an atlas of central complex connectivity using these techniques was that the source data I collected was too ambiguous to be considered evidence for the complete paths of particular neurons. Some alternative data acquisition techniques that would address this problem are described in the following sections.

MARCM: Mosaic Analysis with a Repressible Cell Marker [Lee and Luo, 2001] is a technique wherein one heat-shocks a developing fly larva at a carefully chosen point in development to randomly trigger a crossing-over between particular chromosomal regions of dividing cells. The genotypes are carefully arranged such that in some proportion of the daughter cells derived from these divisions, the GAL80 suppressor of GAL4 is removed completely, allowing GAL4 to drive a reporter in only a subset of the cells in which it would normally be expressed. Unfortunately, using MARCM is very

time-consuming, both in terms of the time required to breed flies with the right genetics and, due to random variation in the effect of the heat-shock, the numbers of flies that must be dissected.

At a late stage in this project, Dr Joanna Young suggested a mating scheme to produce flies with a genotype suitable for doing MARCM, which I checked with the software described in Appendix C . (The scheme is shown in Figure 117 in that Appendix.) Unfortunately, this work is still at a preliminary stage, and we do not yet have the source stocks required to start the process.

FLP-out: A simpler alternative to the MARCM method for producing more selective images of neurons is the FLP-out technique [Basler and Struhl, 1994], used to good effect in papers such as [Wong et al., 2002]. In this technique a fly of genotype $y\,w\,FL122\,;\,\frac{Sp}{CyO}\,;\,UAS>CD2,y+>CD8::GFP$ is crossed to a GAL4 driver line. A third-instar larva from the progeny of this cross is then heat-shocked, which will cause the $CD2,y+$ cassette to be removed in a random selection of cells. In those cells, CD8::GFP will now be driven wherever there is GAL4 expression. This is a simple, single-generation technique which may produce images with isolated neurons. I initially had no success in using this construct. Eventually, with the help of Prof Gary Struhl, who provided the construct, we concluded that the transgene in our stock must have been lost. He kindly sent a replacement, but by that time we had chosen to abandon this technique in favour of pursuing the algorithmic side of the project with my existing data. Recently Dr Joanna Young has had good results in our laboratory with the replacement construct, so in the future we may wish to pursue this further.

Brainbow: A remarkable recent development in fluorescent labelling of neurons in mice is the "brainbow" technique described in [Livet et al., 2007]. This method creates cells that have a random number of working copies of four fluorescent proteins such that up to 150 distinct colours of neurons can be seen in a single image. Not only are these results astonishing in æsthetic terms, they offer the possibility of collecting connectivity data in bulk for the fly brain, had we a *Drosophila* version of the same technique. (According to [Lichtman et al., 2008], a review of Brainbow's use for collecting connectivity data, there are efforts underway to produce a fruit fly version.) This does not completely solve the problem of ambiguity, of course, since 150 distinguishable colours must be considered in the context of the hundred thousand or so neurons in the fly brain. However, with a subset of neurons selected by a GAL4 driver line, as in this thesis, this number of distinct colours could enable simple

disambiguation of the neurons visible in each scan. This is a very exciting technique, and we can only hope that it will be possible to use it in *Drosophila* soon.

Electron Microscopy: A complete change of approach to the problem of discovering connectivity would be to use image data collected from an electron microscopy (EM) method instead of light-based microscopy. The techniques described in this thesis would not be directly applicable, but the enormously improved resolving power of EM allows one not only to unambiguously distinguish neurons but discover their exact morphology down to spines that would be unresolvable with any light microscope[69]. Recent developments in EM have made the prospect of acquiring imagery of large volumes of the fly brain a realistic prospect, and this is taken seriously as a way of working towards the *Drosophila* "connectome", even though current estimates of the length of time it would take to analyse the data with current technology are in thousands of man-years [Briggman and Denk, 2006, Armstrong and van Hemert, 2009].

Sources of Error: Another way of looking at the issue of data quality that is perhaps more direct is to consider ways in which the accuracy could be improved while still using the same genetics and imaging modality.

section 4.2.2 listed the major effects that might account for differences between registered image stacks of different fly brains. All but the first of these, in fact, contribute to the error in our assessment of morphology of the fly brain as imaged compared to the intact animal prior to dissection. Even if we cannot eliminate these, it is interesting to consider ways in which we could (a) reduce their effect and (b) quantify the error they introduce. For example:

- The increased noise and attenuation of signal at greater depths into the brain can be somewhat compensated for by imaging the brains twice, once from either side of the slide. This may be a simpler alternative to Z-brightness correction and has the virtue that it genuinely collects more information for the deeper parts of the stack.
- I have taken no steps so far to quantify the level of blurring in each axis. It would be interesting to estimate the point spread function of our microscope and use this as input to the deconvolution plugins available for ImageJ.

[69] Of course EM data is not completely free of ambiguity: for example, where neural processes line parallel to the cutting plane there may be problems.

- It is possible that the artefacts from dissection damage could be eliminated in the future by using alternative techniques for preparing the fly for imaging. For example, [McGurk et al., 2007] describes an innovative method for bleaching the cuticle of the fruit fly such that the CNS can be visualized intact (i.e. without dissection) using either OPT (Optical Projection Tomography) or confocal microscopy.

One final point worth mentioning about the images in this thesis is that they were mostly collected before the publication of [Liu et al., 2006], which linked the fan-shaped body to visual learning. In retrospect, the findings in that publication suggest that it may have been worth acquiring multiple images at the same resolution and stitching them together in order to include the optic lobes in the final image, in case the neurons selected by these GAL4 lines also included potential inputs from the visual system of the fly.

7.1.2 Preliminary Work on Neuronal Polarity

Since we are interested in automatically inferring the polarity of the neurons[70] from image stacks, at an early stage of this work I did some tests of several candidate reporters which were preferentially distributed to axons or dendrites. Out of those that I tried,[71] the most promising was the UAS-Dscam[exon17.1]-GFP[72] reporter construct described in [Wang et al., 2004]. This paper demonstrates that of all the many thousand different isoforms of Dscam in *Drosophila*, those spliced with exon 17.1 are predominantly involved in dendritic morphogenesis while those spliced with exon 17.2 are conversely involved in axonal morphogenesis. (Exons 17.1 and 17.2 encode the two possible transmembrane domains of the Dscam isoforms.) Some tests with the UAS-Dscam[exon17.1]-GFP reporter when crossed with c232 (an ellipsoid body GAL4 line with known dendritic and axonal regions [Renn et al., 1999]) showed that when PMT settings were not oversaturating any regions of the stack this showed a clear preference for the dendritic regions. (This caveat with regard to the microscope settings is discussed further under "Contrast in the GFP Channel" below.)

It has been found by other researchers[73] that there are some morphological defects found in brains

[70] i.e. which regions are pre-synaptic (axonal) and post-synaptic (dendritic)

[71] The others were UAS-nod-lacZ, [Clark et al., 1997] UAS-nSyb-GFP and UAS-Syt-HA (a gift from Dr Iain Robinson). The former two produced punctate (and so difficult to interpret) results in isolation. Results from the latter were very poor, most likely due to problems with the anti-HA antibody.

[72] This fly stock was a kind gift from Dr Hong-Sheng Li.

[73] Private communication from Dr Julie Simpson.

which express this UAS-Dscam[exon17.1]-GFP reporter. This strongly suggests that one should be careful about inferring the "dendriteness" of any particular path in my existing image corpus from brains that express this reporter. Thus, rather than attempt to apply results in the dendritically labelled images onto the existing images using registration, I decided to create a combined reporter and deal with these as a distinct set of images. In the future, however, we may be able to develop some way of identifying neurons in each scan based on the location and morphology of the neurons.

Combined UAS-Dscam[exon17.1]-GFP and UAS-lacZ Reporter: Since I obtained the most robust results so far using the cell-filling UAS-lacZ construct, I decided to combine this with the dendritic reporter as opposed, say, to using a membrane marker such as UAS-mCD8::GFP. I devised the mating scheme shown in Figure 118 (Appendix C) to create the UAS-Dscam[exon17.1]-GFP ; UAS-lacZ flies. In the process of checking the correctness of this mating scheme, I developed some computer-based tools for helping with such verification, but as this is a distinct piece of work it is described separately in Appendix C.

The aim of using this combined reporter is to be able to trace neurons using the techniques described in Chapter 5, and then automatically classify the expression as being axonal or dendritic based on the intensity in each channel at that point in the stack. The first results I obtained with this approach were promising (e.g see Figure 115) but demonstrated some important problems. These are summarized in the following paragraphs in this section.

Contrast in the GFP Channel: The most awkward part of acquiring images of brains expressing these two reporters is related to the huge range of levels of fluorescence that can be found in the GFP-expressing channel; this range cannot be represented satisfactorily in 8 bits per voxel. To see why this causes a problem, consider that in order to estimate the "dendriteness" of any point in the image we have to compare the expression levels of UAS-lacZ and UAS-dscam[exon17.1]-GFP. One might hope that a basic rule of thumb such as the following could be the basis of some automated heuristics:

- If there is high GFP expression and no lacZ expression at a given point in the image, that point is likely to be a dendrite.

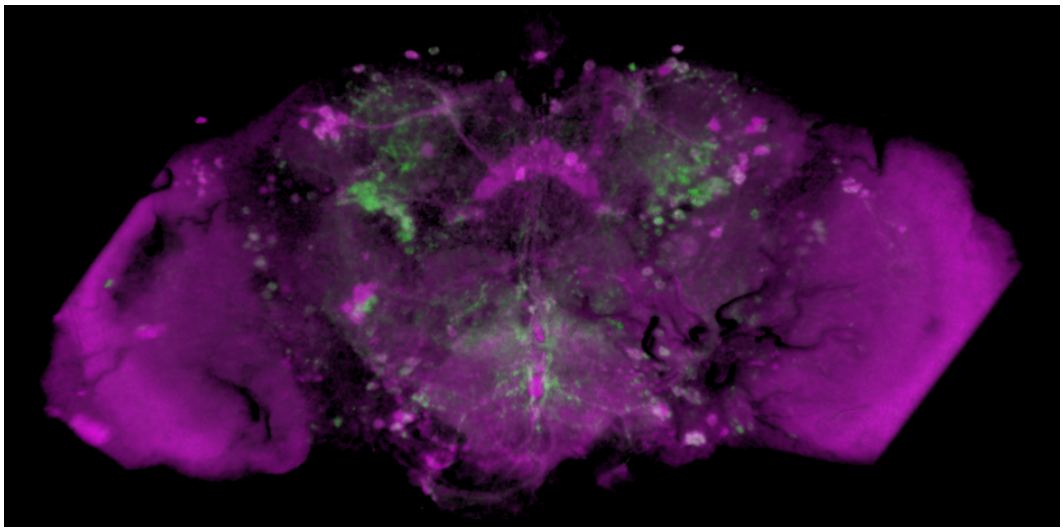


Figure 115 – A volume rendering of an image stack acquired of a c005 x UAS-dscam[exon17.1-GFP];UAS-lacZ fly's brain. The green channel represents expression of UAS-dscam[exon17.1]-GFP (the dendritic reporter) while the magenta channel shows the expression of UAS-lacZ (the cell-filling reporter). This clearly shows a distinction between the dendritic regions of the Fl neurons (lateral to the central complex) and the axonal regions (which notably include the expression in the fan-shaped body).

However, unless particular care is taken in the acquisition of the image stacks, this may not be true. While GFP is highly enriched in dendritic regions of GAL4-positive cells, it is nonetheless expressed throughout each of these cells, so if the gain on the PMT is set too high at acquisition time then the GFP expression in axons may be detected as strongly as that in dendrites. In addition, the lacZ channel may also be low if the process is particularly fine. The control experiments I conducted using the well-characterized c232 GAL4 line demonstrate this point clearly. As you can see in Figure 116, oversaturation of these images (as on the right hand side) risks losing the information that will allow us to infer neural polarity.

Consequently, in my initial set of scans I adjusted the PMT gain in each channel such that a very small proportion of the voxels were saturated. This means that we can have more confidence in determining polarity of processes but it may make fine processes impossible to find, a phenomenon that [Ito et al., 2003] refers to as the "full-moon effect" by analogy with the difficulty of seeing stars that are close to a full moon. The suggestion in that paper is that two scans of each specimen should

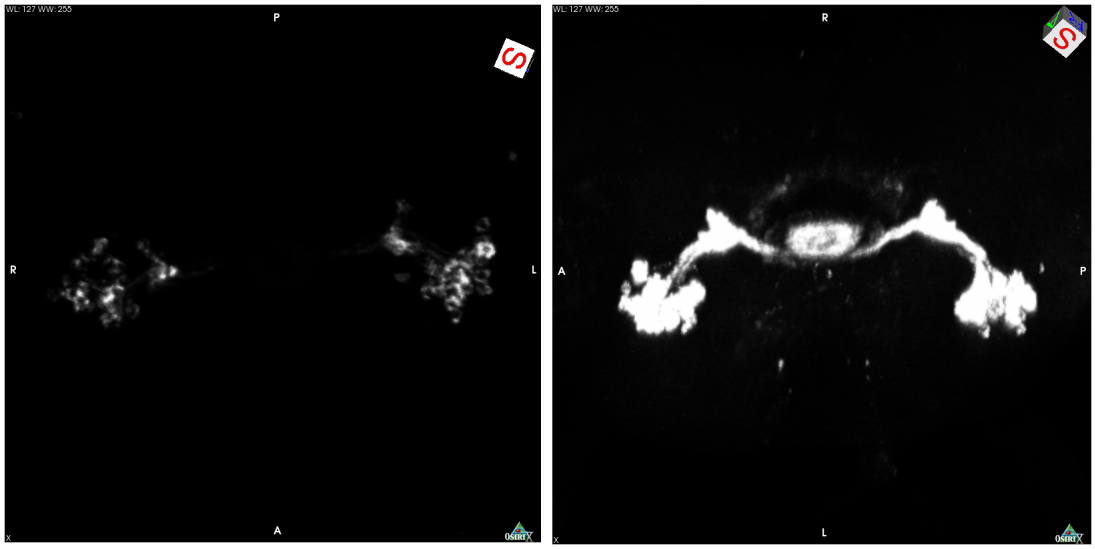


Figure 116 – Both of these images are maximum intensity projections of *UAS-dscam[exon17.1]-GFP; ; c232*. However, in the right-hand image the gain of the PMT was set too high, thus oversaturating the image and losing information. It is harder to use the scan on the right to distinguish the axonal regions (including the ellipsoid body) from the more lateral dendritic regions. [Source images: `c232xdscam-AW.lsm` and `c232xdscam-AQ-top-down.lsm`]

be performed, one "over-exposed" and one "under-exposed".[74] For processing the image stacks here, I have decided to go a step further than this and combine these two stacks into a single stack of 32 bits per channel, in a process inspired by High Dynamic Range image processing in photography. This idea is described in detail below in section 7.1.2, although the implementation is not yet complete.

Z-correction of intensity: Ideally, we would like our heuristics for estimating dendriteness to be independent of the position in the stack at which they are applied. However, as is characteristic in confocal imaging, these stacks have a diminution of the received intensity relative to the source fluorescence as one scans deeper into the tissue.

Typically one would compensate for the attenuation of signal deeper in the stack by using the automatic Z-correction feature of the microscope, which, in the case of our group's confocal microscope, allows one to set different PMT settings at two depths in the brain, and then interpolates the settings for each slice in between. In order to correct for the attenuation we need to model it in some way, and this would only be complicated by using the microscope's automatic Z-correction feature. So, in order to keep the later analysis as simple as possible, I chose to scan with each channel's PMT settings fixed throughout the entire stack.

An idea which I would like to investigate at some point is whether it would be possible to compensate for the variation in intensity and noise through the stack by marking regions of interest which are purely (non-zero) background on slices at different depths.

Offset From Platform Movement: When taking multiple images of the same brain, and with such small sample spacing between the voxels, it is very easy for the stack to be moved slightly out of alignment between one scan and the next. This may be caused, for example, by vibrations in the room or small errors in the mechanics of the microscope. Fortunately, these offsets are near-rigid and are between images of the same subject, so correcting for them is not so complex as the registration tasks considered in Chapter 4. However, I have not yet dealt with this problem in a systematic way for these images.

[74] I shall avoid using the terms "over-exposed" and "under-exposed" when referring to confocal microscopy, since they are somewhat misleading in that context: when using the microscope we do not adjust the detail that can be seen for each fluorescence range by adjusting exposure time, and there is no obvious analogy for the microscope's "amplifier offset" setting in classical photography, at least at the time of acquisition.

No Absolute Measure of Intensity: In previous chapters of this thesis I have not been concerned with the absolute levels of fluorescence from any point in the brain: the concern was simply to detect as many neuronal processes as possible, and this can most simply be done by allowing areas of high intensity to saturate. This loses information about the intensity of fluorescence in the region of cell bodies, for example, but they were considered structures of less interest than the fine and typically much fainter processes extending from the cell bodies. However, for these experiments in estimating dendriteness, it may be useful to have an idea of the absolute measure of fluorescence, such that its range can both be accurately described for individual brains and also allow comparison to other samples with the same genetics; the variation of fluorescence between brains would be interesting to look at.

A possible solution to this would be to add fluorescent microspheres to the preparation,[75] which could be used to provide baseline known values of fluorescence within the image's stack. In the future it would be interesting to see whether the addition of such microspheres would make the later analysis simpler.

An alternative approach which would work with the existing scans is to extract the PMT settings from the LSM file and attempt to estimate the original fluorescence from them using data from Zeiss about the properties of the PMT and amplifier. We could validate results from such calculations by comparing the multiple "exposures" of the same brain, with the understanding that there will also be some bleaching between these exposures.

The four problems listed above are all interesting, but the first (the high contrast in the GFP channel) is the one I have done most work towards so far. The process of combining stacks to compensate for this is described in the next section.

Creating High Dynamic Range Image Stacks [Debevec and Malik, 1997] describes a technique for combining differently exposed photographs of the same scene into a single image of greater bit-depth that can represent a much greater range of intensities. This technique, combined with various types of post-processing[76] to produce the final image, has become popular in conventional photography as a method of creating images of high contrast scenes which still have detail in darker

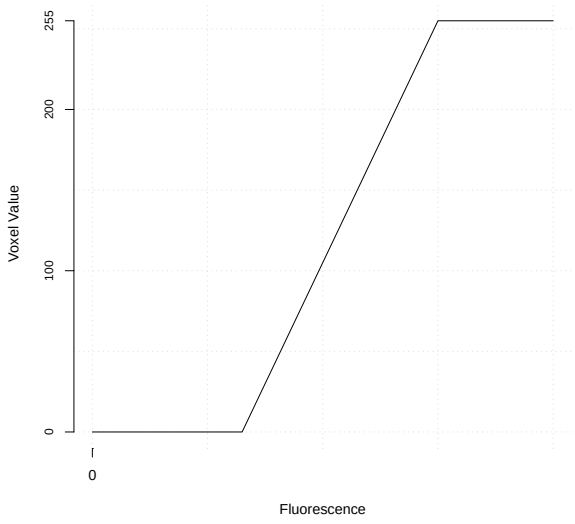
[75] A kind suggestion from Benjamin Schmid.

[76] For example, tone-mapping to reduce the photo to an 8 bit per channel image that can be displayed on computer monitors, which gives images produced using this technique a distinctive unearthly quality.

regions of the image. We can apply the same general idea to confocal image stacks acquired with different detector settings, in order to combine our multiple two-channel stacks into a two-channel 32-bit stack with much higher contrast. The reasons we have to take a different approach from the standard HDR exposure-blending technique are:

1. Debevec et al.'s method uses the exposure time as a known variable for each source image.
2. The response of the PMT for each channel should be approximately linear when the voxel value is not at the extremes of the range [0,255] whereas the corresponding function for luminance to pixel value in photos typically is not.
3. We can use a different amplifier offset for acquiring the image stack with information about the bright regions. For example, we could set the zero voxel value to exclude darker data: this is not possible with conventional photography.

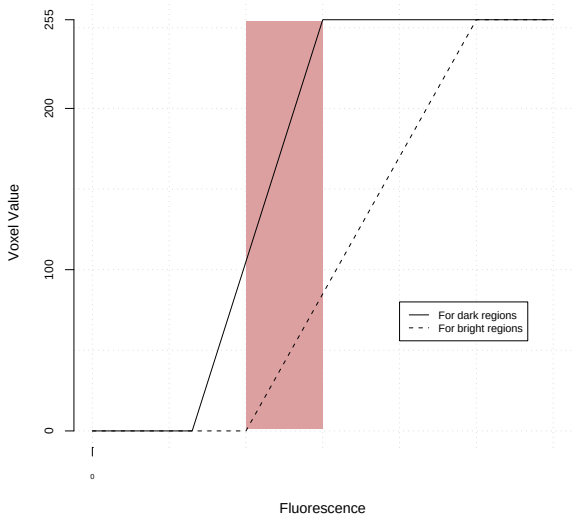
The implication of the second point is that we can model the mapping from luminance (fluorescence at a point) to pixel value as a piecewise linear function like that shown below. (This is just a schematic representation, so units of fluorescence on the x-axis would be arbitrary.)



The fluorescence value at the minimum point of the rising linear region is determined by the "Amplifier Offset" setting of the confocal microscope while the gradient of the rising linear region is determined

by the "Detector Gain" setting. For the acquisition of these images I have kept the "Amplifier Gain" and the LASER power constant.

When we acquire two image stacks of the same subject, we are applying two such detectors, perhaps like this:



The pink shaded region of this graph shows the range of levels of fluorescence which, in this ideal model, will be neither over-saturated nor under-saturated in both images.

The detector gain and amplifier offset values are recorded in the LSM files that are saved by the confocal microscope so it may be possible to infer these graphs solely from those values, with suitable information from the microscope manufacturer, as mentioned above. However, it would be useful to not have to rely on these data being available; in addition, if we combine the images without such data, we can check our results against them later.

The paired values in the fluorescence range where both images have a linear response, which can be clearly seen in a 2D histogram of these values, allows us to estimate the properties of the amplifier in each image and combine the data into a single stack of higher dynamic range than either of the source images. A simple prototype of this technique has been promising, but in the current version can create sharp edges between regions where data is estimated from one and two images.

7.2 Summary and Conclusions

When starting this project there was no free software available to help to annotate neuronal connectivity in 3D image stacks, only expensive and closed-source solutions - the development of the Simple Neurite Tracer tool was driven by this clear need. Since publishing this online I have had feedback from users who have been successfully using this tool for a variety of applications, including tracing non-neuronal structures such as blood vessels. Since the source code is freely available for anyone to alter and develop, both the Simple Neurite Tracer plugin and the standalone Tubeness plugin provide an easy platform for other researchers to try out alternative algorithms with, and I have had a number of enquiries about doing so. Of course, there is a great deal of scope for improving the algorithms used, but as it stands this has been shown to be a capable tool for semi-automatically marking out the paths and topology of neuronal structures. The preliminary work on fully automatic tracing has also been interesting in terms of generating hypotheses about connectivity, although it still needs a richer model of types of expression thoughout the image and further development work to be useful to typical users.

An interesting development that was made public after much of the work in this project was complete was the launch of the DIADEM challenge[77], a competition to encourage development of fully automatic neuron tracing software, with one of the requirements being that the project should be available under a free software license. This is a clear indication that the research community sees work such as that in this thesis as crucial to discovering high quality neuronal connectivity information.

3D image registration is still a bottleneck for many biological image processing algorithms, but we have found that the results from CMTK are of very good quality for this application, even in the slightly unfair cases where the source image data show signs of some damage. An important change since the start of this work was that this tool has been generously released with source code under the GNU GPL, which in the future makes it possible for us to deploy it both in Fiji (with the binaries being called from Java) and on the server side. A particularly attractive possibility in the latter case is to attempt registrations of scans uploaded to our confocal image archive as a matter of course. While in many cases these registrations may fail, if even a small proportion succeed this would save valuable time, and make the technology accessible to users who are not accustomed to running long

[77] `http://diademchallenge.org/`

registration jobs on their desktop or laptop computer. I believe that some of the biggest challenges with regard to image registration at the moment are to make them more accessible to typical users in terms of user interface; something akin to a wizard that guides a user through the process of choosing and applying a particular registration algorithm would be an invaluable tool.

The web-based confocal scan archive described in Chapter 3 provides a simple, distributed platform for archiving both the original image data and annotations. Furthermore, it lets us selectively publish the data either just within a research group or to the world at large. Since the back-end is based on Fiji, it is easy for developers to extend the site to perform more complex image processing tasks. This system has been designed to allow the back-end processing to be distributed over many computers to allow for easy scalability if these tasks are computationally demanding.

In the latter part of Chapter 5, I performed some basic analyses of the annotated traces. In most cases this was done by picking a primary axon of a known neuron type and plotting the deviation from this of similar axons in other brains. In the case of the line 210y, however, I picked a segment of a neuron which we hypothesized may pass information between the fan-shaped body and mushroom bodies. Although it seems likely, in the light of other literature, that this process does not synapse in the mushroom bodies, this demonstrates the power of this technique for accurately describing such posited connections in a testable way. Without the original source data (published via the confocal scan archive) and the voxel-by-voxel traces (produced with Simple Neurite Tracer) we could not describe such connections sufficiently accurately for them to be identifiable with other neurons. Chapter 6 demonstrates a distinct type of computational neuroanatomy, where the analysis is based on comparing the volumetric data of the registered scans rather than the neuron traces. This suggests a method for describing the expression within the fan-shaped body based on a fine-grained profile of expression by layer, as opposed to the currently ambiguous terms “F1” and “F5” used in the literature.

A further contribution of this work, which is evident from the images in the results sections of Chapter 5, is that the Simple Neurite Tracer and the ImageJ 3D viewer enables users to create publication quality images of neuronal tracings both with surface renderings of neuronal structures and volume renderings of the original data. This aspect of the tool is also important in everyday use, where easily manipulable 3D views of data are far more easy to understand than either the slice-by-slice presentation or orthogonal projections offered by many free software solutions.

In summary, this work represents a significant contribution to practical methods for generating connectivity, morphology and topology data for neurons from images of the fruit fly brain, and this has been demonstrated with case studies based on data collected for the central complex of *Drosophila melanogaster.* This field is rapidly gaining momentum as such techniques are crucial to generating bulk connectivity data, and in the future we can look forward to this being the basis for simulation and modelling of ever more complex circuits in the fly brain.

8 Acknowledgements

I owe a great debt of gratitude to my supervisors, Dr Douglas Armstrong and Dr Dean Baker for their patient advice, support, and help throughout this work. I was very fortunate to be able to work with members of the Armstrong group and Jarman group for much of this project, who were always unfailingly helpful, encouraging and fun - in particular, I would like to thank Dr Joanna Young, who has been enormously helpful on matters of fruit fly neuroanatomy. In addition, Dr Bilal Malik, Jane Ewins, Dr Petra zur Lage, Dr Lynn Powell, Sadie Kemp, Dr Sebastian Cachero, Dr Raphaela Kitson-Pantano, Prof Andrew Jarman and Seymour Knowles-Barley all contributed greatly to making my lab work as safe and enjoyable as possible, and provided valuable feedback on the various projects presented in this thesis. Seymour Knowles-Barley was also an excellent collaborator on the online confocal archive project discussed in Chapter 3. It has frequently been very helpful to talk about my work with members of the Institute of Adaptive and Neural Computation, in particular Dr David Sterrat.

At an early stage of this project, my supervisors encouraged me to visit Prof Martin Heisenberg's group in Würzburg, which turned out to be a fantastic experience - even beyond their kind hospitality, the advice, enthusiasm and openness of Dr Johannes Schindelin and Dr Arnim Jenett changed the direction of my PhD work in a very positive direction. As a result, I was lucky enough to be introduced to many remarkable mathematicians, software developers and biologists - in particular I would like to mention those who participated in the "hackathons" generously hosted by Janelia Farm (a research campus of the Howard Hughes Medical Institute) and the Max Planck Institute for Cell Biology and Genetics in Dresden. Working with Dr Albert Cardona, Stephan Saalfeld, Stephan Preibisch, Dr Jean-Yves Tinevez, Dr Gregory Jefferis, Benjamin Schmid, Dr Johannes Schindelin, Dr Torsten Rohlfing, Dr Felix Evers, Dr Daniel White and Dr Pavel Tomacak on Fiji and other projects has been enormously inspiring, and I hope that we will all continue to collaborate in the future. Their willingness to contribute to free software, in particular, meant that I was able to accomplish far more in this PhD than would otherwise have been possible. These hackathon events have been incredibly productive weeks for all of us, and I believe it shows great vision on the part of the hosting institutions that they are willing to support them.

Since so much of this work is based on ImageJ, it would be remiss not to thank Wayne Rasband, whose enormous amount of work in developing the project has been of such benefit to so many of us

in the biological image processing community.

In the first year of this project I was given some excellent advice and guidance by Prof Richard Baldock and Dr Duncan Davidson which helped greatly to keep this work on track, and I would like to thank them both for their time and assistance.

It has been very exciting to be a part the worldwide community of *Drosophila melanogaster* researchers for these years. In particular, I have been consistently delighted that so many distinguished academics in the field have taken the time to give me advice and encouragement at conferences, during stressful poster sessions and so on. In particular, the words of Prof Martin Heisenberg, Prof Nick Strausfeld, Prof Gene Myers, Prof Uwe Homberg, Dr Gregory Jefferis, Dr Julie Simpson and Dr Cahir O'Kane at various points have been very influential for me. The work in this thesis was also heavily dependent on the gifts of fly stocks from various labs, which are acknowledged through the text.

In the course of developing the software in this project I was greatly helped by the various users and developers who were willing to try early versions of software and give me constructive bug reports and suggestions. As well as the Fiji developers mentioned above, Adrianna Teriakidis, Dr Benny Lam and Dr Ting Zhao were kind enough to give me feedback on early versions of the software, from bug reports through to suggestions for algorithmic improvements.

This work was made possible by funding from the Doctoral Training Centre in Neuroinformatics, hosted by the Institute of Adaptive and Neural Computation at the University of Edinburgh, which is funded and supported by the Life Sciences Interface programme of the EPSRC, cofounded with the MRC and BBSRC. I am grateful to all of those organizations for their support and vision in setting up such a remarkable programme for PhD students. In particular, I would like to thank Pat Ferguson and the directors of the DTC, Prof David Willshaw, Dr Mark van Rossum and Dr Jim Bednar, for their tireless work on the programme.

Lastly, I doubt that I would have persisted in this work if it were not for the amazing care and support I have had from my partner, my family and wonderful friends. I owe them all more than I can say.

9 Declaration of Authorship

In accordance with regulation 2.5 of the University of Edinburgh Postgraduate (Research) Assessment Regulations, I hereby declare that this thesis was composed by myself and the work that it describes was carried out by myself except where clearly indicated in the text. I also declare that this work has not been submitted for any other degree or professional qualification.

Mark Longair

10 Appendix A: Confocal Database API

The web-based confocal scan archive described in Chapter 3 has a simple API for interrogating the database and uploading scans and annotation files. This is not yet completely implemented, but the documentation for all the planned API calls can be found below.

10.0.1 HTTP-based API

One of the requirements for our system is that it should be simple to interrogate programmatically, in order to enable students to have a simple way of accessing images and their metadata for short projects. An HTTP-based API in this case makes the most sense, since it can be implemented on the server side as any other CGI script and does not require special measures to make it accessible (e.g. opening additional ports in firewalls, etc.) Fortunately the data that we need to return from the database is very simply structured so we could provide an interface using virtually any of the various possible remote procedure call protocols. The decision of which of these to use is not particularly interesting in this case, so I have implemented the server side API functionality as a set of functions which all take a hash of parameters and return a table as results. This underlying functionality is then made accessible by a small CGI script (confocal/archive/api-csv) that accepts input as HTTP POST parameters and returns results as the body of the HTTP response in CSV format, specified in RFC 4180. If a SOAP API is required, for example, it would be simple to create a new script based on api-csv that implements that protocol.

The following methods are provided by the API. (Only quite basic functionality is accessible via these methods, but more methods can be simply added on request.)

- **login**
  - Inputs (required): *username* : string, *password* : string
  - On success, returns a "ticket" that must be used in subsequent calls:
    *

result	ticket
success	<*ticket-value*>

  - On failure, returns:
    *

result	message
error	<*error-message*>

- **logout**
  - Inputs (required): *ticket* : string
  - On success, returns:
    *

result
success

  - On failure, returns:
    *

result	message
error	<*error-message*>

- **get-scans**
  - Inputs (required): *ticket* : string
  - Inputs (optional): *user* : string, *md5sum* : string, *id* : string, *line_id* : string
  - On success, returns information for each matching string:
    *

s.id	s.date	s.date	s.md5sum	... etc., see below
<*scan-id*>	<*scan-date*>	<*scan-date*>	<*scan-md5sum*>	... etc.
...	...	...	...	...

    * The full list of columns, with their meanings, follows:
      · "s.id": The scan's ID
      · "s.date": The date on which the scan was uploaded
      · "s.md5sum": The MD5sum of the scan,
      · "s.username": The username of whoever uploaded the scan
      · "s.name": The displayed name of the scan, which may be changed by the owner
      · "s.description": An option longer description of the scan
      · "s.first_uploaded": A full timestamp of when the scan was uploaded
      · "s.width": the width of the scan in samples (i.e. an integer value)
      · "s.height": the height of the scan in samples (i.e. an integer value)
      · "s.depth": the size of the image stack in samples (i.e. an integer value)
      · "s.channels": the number of channels in the image
      · "s.notes": Additional notes on the scan, set by the scan's owner

· "s.spacing_x", "s.spacing_y", "s.spacing_z": the separation of samples in *x, y* and *z*

· "s.spacing_units": The units of "s.spacing_x", "s.spacing_y" and "s.spacing_z"

· "s.file_size": The file's size in bytes

· "s.loader": The ImageJ loader that successfully could load the file

· "s.imagej_type": The name of the constant in ImageJ's ImagePlus class that describes the type of the image, one of "GRAY8", "COLOR_256", "GRAY16", "GRAY32" or "COLOR_RGB"

· "s.original_filename": What was supplied as the original filename on upload

· "s.line_id": The ID of the genetic line that the scan is associated with

· "l.name", "l.comments", "l.gene": For convenience, the name, comments and gene associated with that line_id

– On failure, returns:

result	message
error	<*error-message*>

- **get-lines**

  – Inputs (required): *ticket* : string

  – Inputs (optional): *line_id* : string, *line_name* : string

  – On success, returns information for each matching line:

id	name	comments	gene
<*line-id*>	<*line-name*>	<*line-comments*>	<*line-gene*>
...	...	...	...

  – On failure, returns:

result	message
error	<*error-message*>

- **set-line**

  – Inputs (required): *ticket* : string, *scan_id* : string, *line_id* : string

  – On success, returns:

result
success

- On failure, returns:
  *

result	message
error	<*error-message*>

- **upload-scan, get-landmark-tags, get-channel-tags, get-scan-tags, get-annotation-file, get-annotation-file-list, upload-annotation-file, get-lsm-properties**
  - These methods are only stubs on the server at the moment, and are not currently implemented.

11 Appendix B: Simple Neurite Tracer .traces File Format

The `.traces` files that are saved by Simple Neurite Tracer are gzip compressed XML. The plugin will also load uncompressed XML files, but by default they are saved in the compressed form.

The XML DTD is included in the DOCTYPE of each file. The root element is always <tracings>, and this can contain the following elements:

- <imagesize>
- <samplespacing>
- <path>
- <fill>
- <node>

<imagesize>

There must be exactly one of these elements present, with attributes that describe the size of the image in terms of number of voxels across, up and down, e.g.:

```
<imagesize width="520" height="434" depth="117"/>
```

<samplespacing>

There must be exactly one of these elements present, with attributes that describe the spacing of the samples in the image (voxels) in world-coordinates, e.g.:

```
<samplespacing x="0.28838738962693633"
               y="0.28838738962693633"
               z="1.2"
               units="micrometers"/>
```

<path>

The `<path>` element can have the following attributes:

- `id`: a non-negative integer ID unique among the `<path>`s in this file
- `startson`: if this is present, it gives the ID of the path which the beginning of this path branches off from. If `startson` is specified, then either the deprecated attribute `startsindex` or the recommended attributes `startsx`, `startsy` `startsz` must be specified as well.
- [**deprecated**] `startsindex`: This attribute used to indicate the 0-based index of the point in the other Path where the branch occurred. Please use `startsx`, `startsy` and `startsz` instead.
- `startsx`, `startsy` and `startsz`: These attributes indicate where on the path specified by `startson` the branch occurs. If one of these is attributes is specified, all must be specified.
- `endson`: if this is present, it gives the ID of the path which the branch ends on. If `endson` is specified, then either the deprecated attribute `endsindex` or the recommended attributes `endsx`, `endsy` `endsz` must be specified as well.
- [**deprecated**] `endsindex`: This attribute used to indicate the 0-based index of the point in the other Path where this path joins it. Please use `endsx`, `endsy` and `endsz` instead.
- `endsx`, `endsy` and `endsz`: These attributes indicate where on the path specified by `endson` this path ends. If one of these is attributes is specified, all must be specified.
- `name`: A string giving the name of this path
- `reallength`: The length of this path found by summing the Euclidean distance between each consecutive pair of points, in the units specified in <samplespacing>
- `fitted`: If present, this attribute gives the ID of another path which is a version of this path after the centre-line has been adjusted and the radius at each point found. If this attribute is present, the `fittedversionof` attribute may not be.
- `fittedversionof`: If present, this attribute gives the ID of another path which was the source version for this one. Typically the path specified does not have radii defined for each point, although this is not always the case. If this attribute is present, the `fitted` attribute may not be.

- `usefitted`: This attribute must be present if either the `fitted` or `fittedversionof` attributes are. This attribute is either `"true"` or `"false"`. It should only be "true" for paths that have a fitted version, when it implies that the user wants the fitted path to be display in favour of the unfitted one. If "false" and this path has a fitted version, it means that this path should not be displayed. It should always be "false" for paths that are fitted versions of other paths. **Note:** this is confusing and regrettable; in later versions this will be replaced by attributes with simpler semantics.

The <path> element may contain zero or more <point> elements. These are described below:

<point>

This represents a point in a path. A point element may have the following attributes:

- `xd`, `yd`, `zd`: These three attributes give the position of the point in world coordinates. e.g. you can use these coordinates directly to calculate the length of paths.
- [**deprecated**] `x`, `y`, `z`: These attributes represent the position of the point in image coordinates (i.e. indices of voxels in each axis). They are still generated for backwards compatibility, but it is better to use `xd`, `yd` and `zd`.
- `r`: If present, this attribute gives the radius of the neuron at that point.
- `tx`, `ty`, `tz`: If present, these attributes give the tangent vector along the neuron at (`xd`, `yd`, `zd`)

<fill>

The `<fill>` element represents a fill around a path. It contains all the points found in the search starting from points on the path, but those that actually make up the fill are only those below the threshold specified in the attributes. (This is so that the search can be restarted if the fill is reloaded.) The `<fill>` element can have the following attributes:

- `id`: The ID of the fill, a non-negative integer unique among all the other fill IDs in this file.
- `frompaths`: A comma (+ optional space) separated IDs of the paths from which this fill started. e.g. if this attribute is `frompaths="2, 0"` then there are nodes with distance 0 at each of the points on `<path id="2" ...` and `<path id="0" ...` in this fill.

- `metric`: this can either be `reciprocal-intensity-scaled` or `256-minus-intensity-scaled`. The former means that the cost of moving to a point from an adjacent one is the Euclidean distance between the two divided by the intensity value at the latter. The former means that the cost of moving to a point from an adjacent one is 256 minus the intensity value at the latter point all multiplied by the Euclidean distance between the two. **Note:** `metric` is a bad name for this attribute since these are not metrics in the strict sense: for example, they are not symmetric.

- `threshold`: all the points with a "distance" less than this path are considered to be part of the fill.

<node>

- `id`: Each node in the search has a non-negative integer ID which is unique within the enclosing fill.

- `x`, `y`, `z`: the position of the node in the image stack in image co-ordinates, i.e. 0-based indices in voxels.

- `previousid`: If present, this ID gives you the previous node on the shortest route from the original paths to this point. It is not present for the points on the original paths, which also have a `distance` attribute equal to 0.

- `distance`: This is the minimum "distance" so far found for any route moving from any point on the original paths to this node. (The complete route can be reconstructed by following `previousid`s.)

- `status`: this attribute can either have the value `open` or `closed`, which have their conventional meanings in A* search.

12 Appendix C: Planning Mating Schemes

An important skill for *Drosophila* researchers is being able to plan mating schemes in order to produce flies with particular genetics, starting from a limited set of source stocks. There is a good introduction to this process in [Greenspan, 1997], which particularly emphasizes the use of balancer chromosomes to simplify the planning: these are chromosomes with such severe inversions (or other mutations) that recombination events are very rare. Nonetheless, for a novice Drosophilist the possibilities are daunting - mixing flies of two genotypes can create a bewildering array of different genotypes in the progeny, and since it may take weeks to discover any errors, the consequences for getting the scheme wrong can be serious. Two example mating schemes, relevant to the Further Work section of Chapter 7, are shown in Figures 117 and 118.

One insight that particularly suggested that this is an area where computational tools may be of use is that planning these mating schemes is analogous to the process of finding a mathematical proof, where:

- Source stocks are axioms.
- The operations of selection (based on phenotype or gender) and mating are rules that can produce new theorems.
- The desired final genotype is the theorem to prove.

This suggests the possibility of two useful computer-based tools:

1. A system which allows one to simulate mating schemes, choosing the operations manually, in order to check that the scheme will indeed produce flies of the right genotype.
2. A system which will attempt to find the "best"[78] mating scheme to produce a fly with particular genetics from a given set of source stocks.

In both cases, the development of such tools can be made much simpler by regarding any recombination event as an error and disallowing operations where it might have an effect. For simple mating schemes

[78] Which scheme is "best" might be based on a heuristic which primarily considers the number of generations required, then how easy it is to distinguish phenotypes for selection and possibly other factors such as how healthy each generation is likely to be.

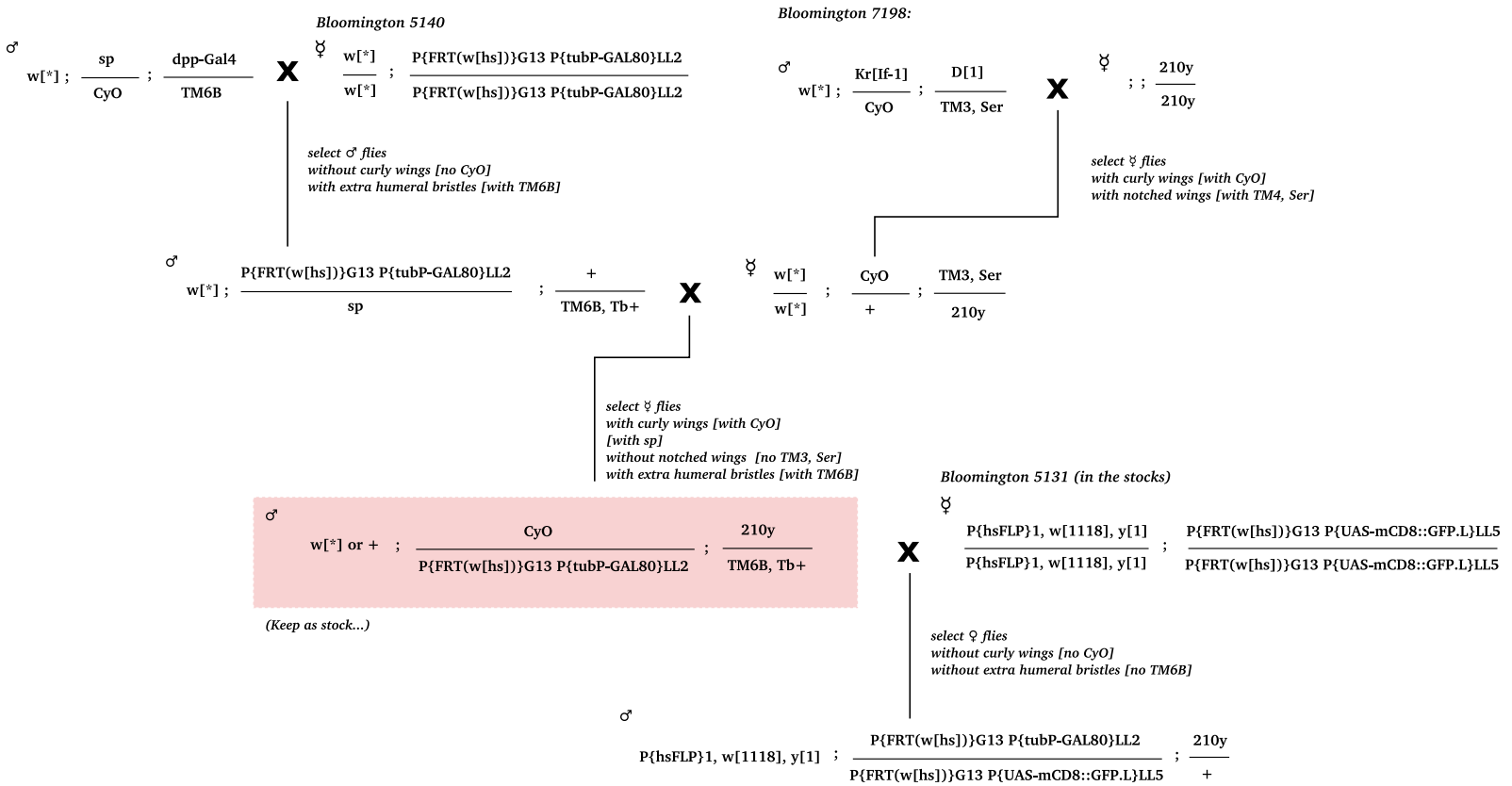


Figure 117 – Suggested mating scheme for MARCM analysis of 210y flies.

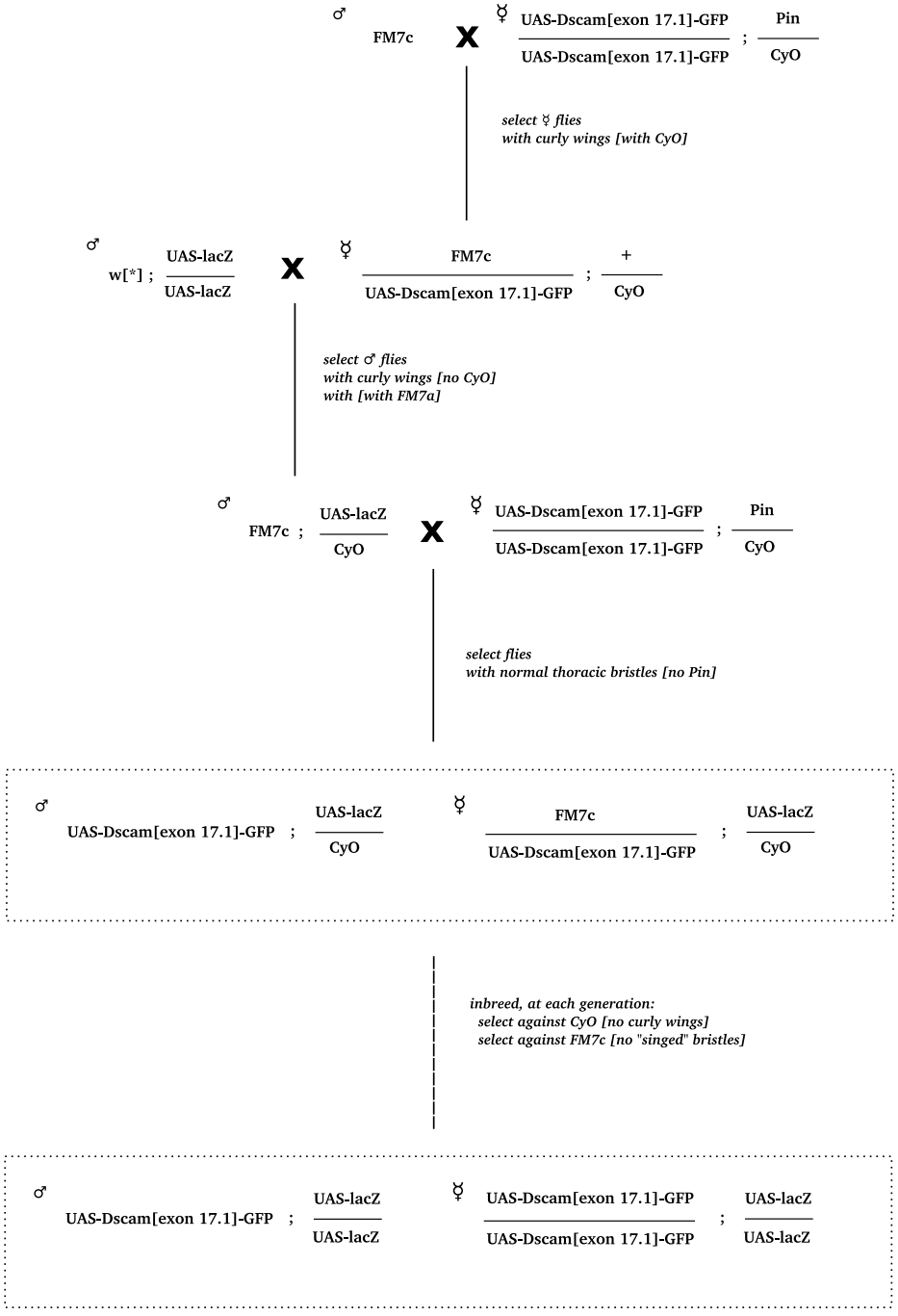


Figure 118 – The mating scheme used to generate stock UAS-Dscam[exon17.1]-GFP;UAS-lacZ flies.

this is likely to be what is wanted, although feedback from demonstrations of my prototype system suggested that it would be a great feature to allow recombination and produce statistics about the relative proportions of particular genotypes in the offspring of crosses. However, this simplifying assumption means that the problem can be reduced to one of first-order logic, and I have done some preliminary work with Dr Graham Steel on formalizing this such that automated theorem-proving tools could be used to find mating schemes. This work is far from complete at the moment, but is a potentially very interesting application of mathematical logic to a biological problem.

On the other hand, I have done enough preliminary work on the first project to produce an early version of a *Drosophila* stock management website that also allows some simple mating scheme checking. This uses a simplified model of *Drosophila* genetics which can be approximated by the following definitions:

- A *mutation* has a set of *phenotypes*.
- A *mutation* can contain other *mutations*. (This is so that we can represent, for example, a complicated balancer mutation as a single entity while reusing the marker mutations which they also contain.)
- A *phenotype* has descriptions of the homozygous / heterozygous effect in males / females for a particular stage of development and body part, if applicable.[79]
- A *mutation* can be marked as homozygous lethal / sterile in males / females.
- A *mutation* can be marked as a balancer, which just means in this context that it will not allow recombination between any chromosome that contains it and any other.
- A *mutation* may be specific to a chromosome, (1 to 4) or the chromosome may be set to 0 to indicate that this is a transgene.
- A *chromosome* has a set of *mutations* and a number (1 to 4) indicating which chromosome it is.
- A *fly_genotype* has many *chromosomes* and a sex.[80]

[79] It is useful to include the body part as distinct from the free text description so that we can warn the user that scoring two different phenotypes in the same body part may be difficult; for example, it is difficult to score any other bristle phenotype in the presence of *singed*.

[80] This can be calculated from the number of X chromosomes, of course, so is actually redundant.

- A *fly_genotype* may optionally be in a *vial*.
- A *vial* can have many *fly_genotypes*.
- A *stock_collection* has a name and many *vials*.

Ultimately, I would want to use information from Flybase about mutations, phenotypes and genotypes to populate this database, but for the demonstration web application I have entered some such information manually. This can be seen in Figure 122.

A user can create a mating scheme and populate it with a number of source vials (Figure 119). Then they are offered a choice of options for the next step, as shown in Figure 120. Any of these options produces a new "working vial", which can be used in subsequent operations as well - these are shown with different coloured backgrounds, as shown in Figure 121.

This prototype is sufficient for basic checking of mating schemes, although there are still many improvements that would need to be made to it before releasing a production version, mostly to improve the speed of response. For example, it can currently take many seconds to calculate what the progeny of a cross will be and display them. In addition, there are some missing features that I would like to implement, in particular the ability to produce well-formatted, printable output similar to the diagrams I manually created for Figures 117 and 118.

There is a great appeal, I believe, in this novel melding of stock management software and mating scheme planning. I hope that it will be possible to take this work forward in the future.

Source Vial C (Remove from Scheme)

Fly's sex is **male**

Genotype is:

$$\frac{w[*]}{/} \; ; \; \frac{Kr[If\text{-}1]}{CyO} \; ; \; \frac{D}{Tb[+],\ TM6B} \; ; \; \frac{+}{+}$$

Phenotypes:

wings curl up at the ends
wings extended like jet plane instead of straight back (Greenspan)
Eyes are small (less than half size) and rough
Extra bristles on humeral patches

Edit genotype Delete genotype

Fly's sex is **female**

Genotype is:

$$\frac{w[*]}{w[*]} \; ; \; \frac{Kr[If\text{-}1]}{CyO} \; ; \; \frac{D}{Tb[+],\ TM6B} \; ; \; \frac{+}{+}$$

Phenotypes:

wings curl up at the ends
wings extended like jet plane instead of straight back (Greenspan)
Eyes are small (less than half size) and rough
Extra bristles on humeral patches

Edit genotype Delete genotype

Source Vial D (Remove from Scheme)

Fly's sex is **male**

Genotype is:

$$\frac{w[*]}{/} \; ; \; \frac{Kr[If\text{-}1]}{CyO} \; ; \; \frac{D}{Ser,\ Tm3} \; ; \; \frac{+}{+}$$

Phenotypes:

wings curl up at the ends
wings notched
wings extended like jet plane instead of straight back (Greenspan)
Eyes are small (less than half size) and rough

Edit genotype Delete genotype

Fly's sex is **female**

Genotype is:

$$\frac{w[*]}{w[*]} \; ; \; \frac{Kr[If\text{-}1]}{CyO} \; ; \; \frac{D}{Ser,\ Tm3} \; ; \; \frac{+}{+}$$

Phenotypes:

wings curl up at the ends
wings notched
wings extended like jet plane instead of straight back (Greenspan)
Eyes are small (less than half size) and rough

Edit genotype Delete genotype

Figure 119 – Two "source" vials, which might be used to start a mating scheme.

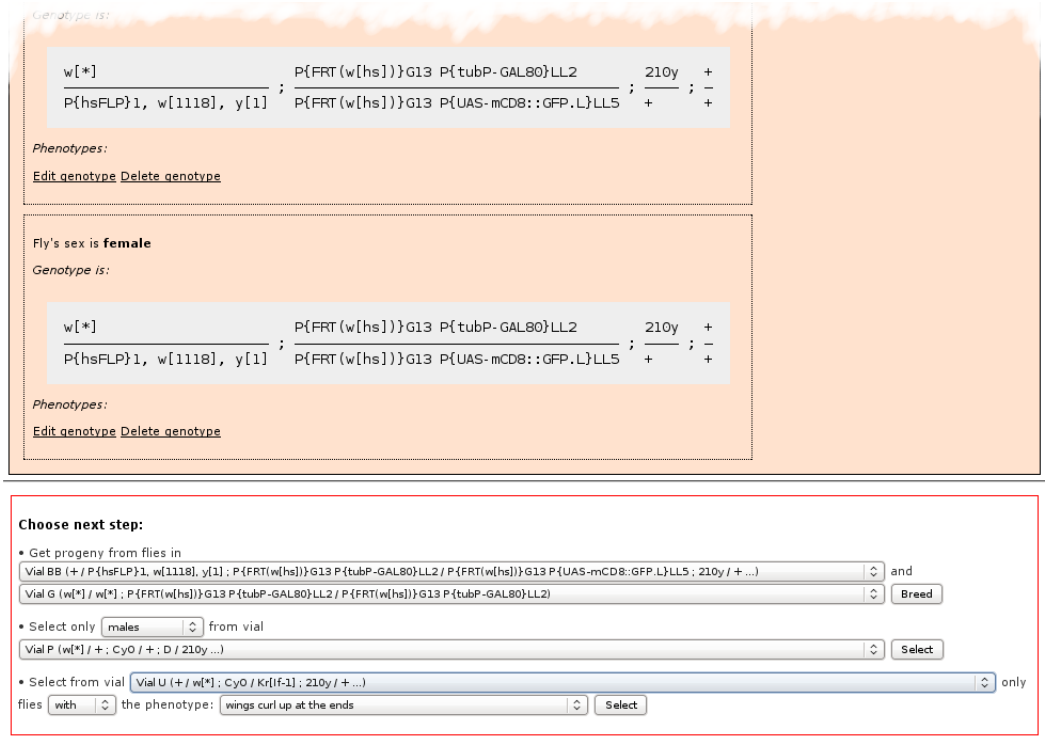


Figure 120 – The "Choose next step" box at the bottom shows the three choices that the user always has: (a) to allow flies from two vials to freely breed, (b) to select flies of a particular sex from a vial or (c) to select for or against a particular phenotype.

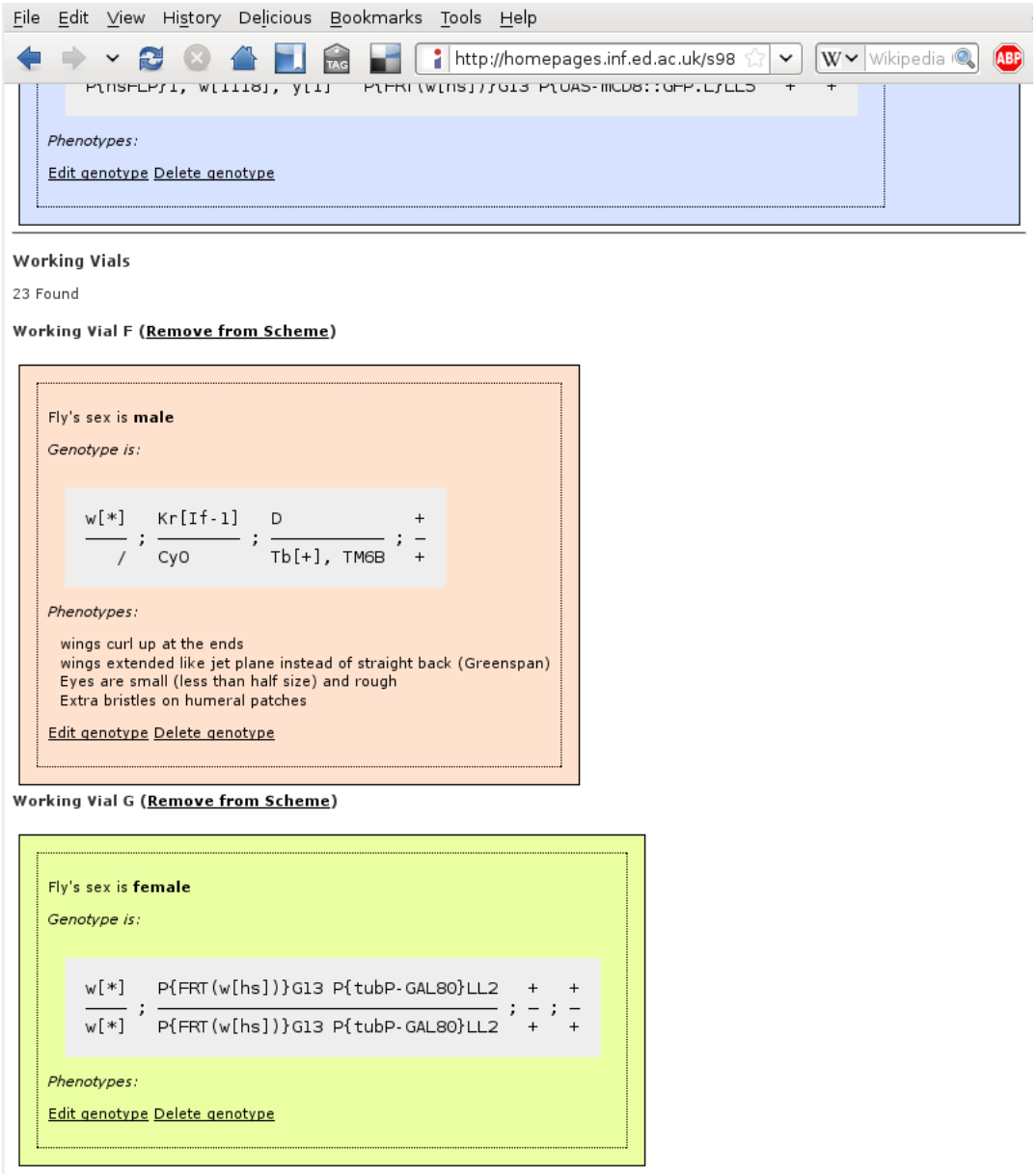


Figure 121 – "Working" vials are shown in different colours to distinguish them from the source vials.

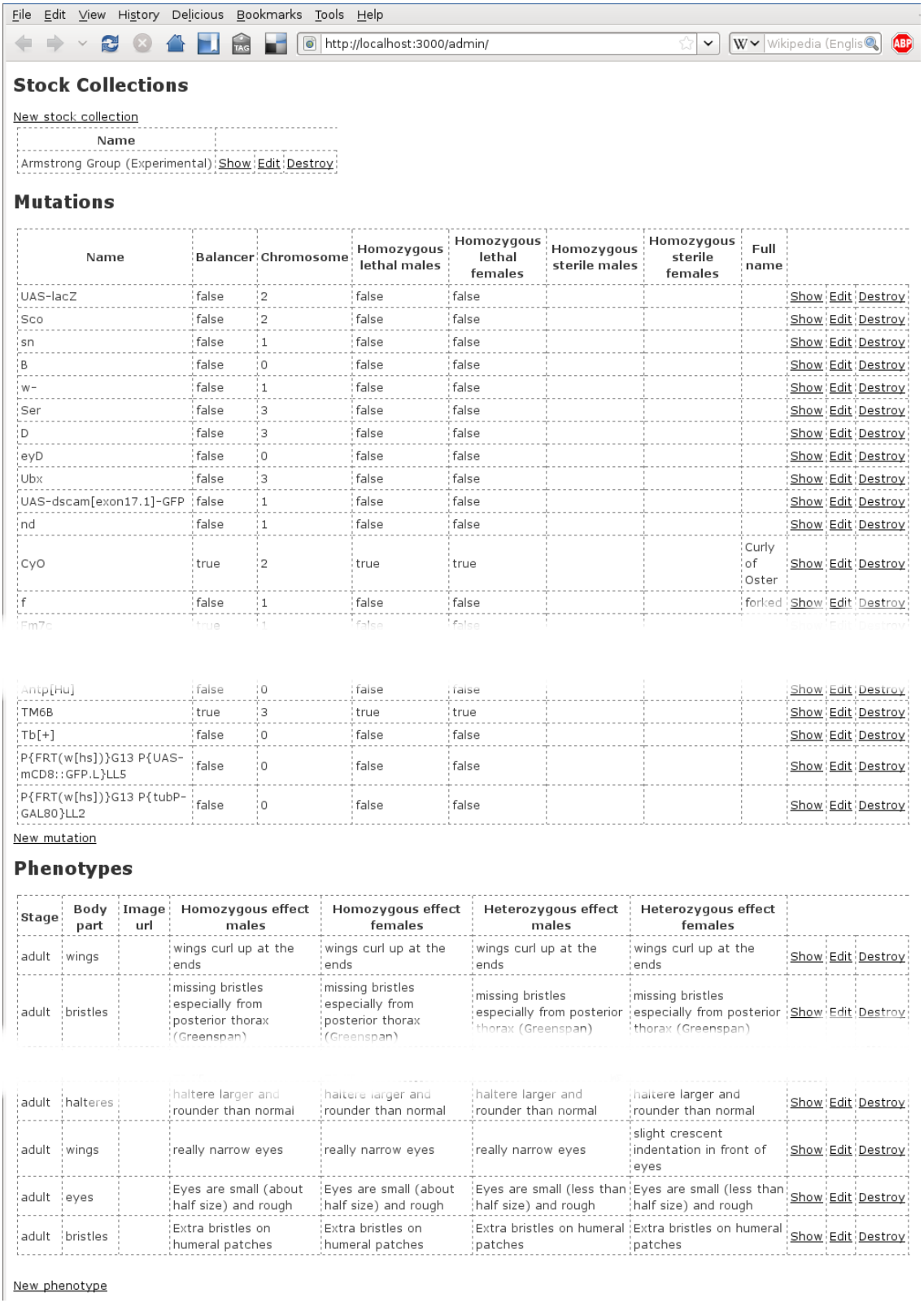


Figure 122 – The administrator's interface, where one can define new mutations and phenotypes. (Some sections of this composite screenshot have been elided to aid readability of the text.)

References

Mark D. Adams, Susan E. Celniker, Robert A. Holt, Cheryl A. Evans, Jeannine D. Gocayne, Peter G. Amanatides, Steven E. Scherer, Peter W. Li, Roger A. Hoskins, Richard F. Galle, Reed A. George, Suzanna E. Lewis, Stephen Richards, Michael Ashburner, Scott N. Henderson, Granger G. Sutton, Jennifer R. Wortman, Mark D. Yandell, Qing Zhang, Lin X. Chen, Rhonda C. Brandon, Yu-Hui C. Rogers, Robert G. Blazej, Mark Champe, Barret D. Pfeiffer, Kenneth H. Wan, Clare Doyle, Evan G. Baxter, Gregg Helt, Catherine R. Nelson, George L. Gabor Miklos, Josep F. Abril, Anna Agbayani, Hui-Jin An, Cynthia Andrews-Pfannkoch, Danita Baldwin, Richard M. Ballew, Anand Basu, James Baxendale, Leyla Bayraktaroglu, Ellen M. Beasley, Karen Y. Beeson, P. V. Benos, Benjamin P. Berman, Deepali Bhandari, Slava Bolshakov, Dana Borkova, Michael R. Botchan, John Bouck, Peter Brokstein, Phillipe Brottier, Kenneth C. Burtis, Dana A. Busam, Heather Butler, Edouard Cadieu, Angela Center, Ishwar Chandra, J. Michael Cherry, Simon Cawley, Carl Dahlke, Lionel B. Davenport, Peter Davies, Beatriz Pablos, Arthur Delcher, Zuoming Deng, Anne D. Mays, Ian Dew, Suzanne M. Dietz, Kristina Dodson, Lisa E. Doup, Michael Downes, Shannon Dugan-Rocha, Boris C. Dunkov, Patrick Dunn, Kenneth J. Durbin, Carlos C. Evangelista, Concepcion Ferraz, Steven Ferriera, Wolfgang Fleischmann, Carl Fosler, Andrei E. Gabrielian, Neha S. Garg, William M. Gelbart, Ken Glasser, Anna Glodek, Fangcheng Gong, J. Harley Gorrell, Zhiping Gu, Ping Guan, Michael Harris, Nomi L. Harris, Damon Harvey, Thomas J. Heiman, Judith R. Hernandez, Jarrett Houck, Damon Hostin, Kathryn A. Houston, Timothy J. Howland, Ming-Hui Wei, Chinyere Ibegwam, Mena Jalali, Francis Kalush, Gary H. Karpen, Zhaoxi Ke, James A. Kennison, Karen A. Ketchum, Bruce E. Kimmel, Chinnappa D. Kodira, Cheryl Kraft, Saul Kravitz, David Kulp, Zhongwu Lai, Paul Lasko, Yiding Lei, Alexander A. Levitsky, Jiayin Li, Zhenya Li, Yong Liang, Xiaoying Lin, Xiangjun Liu, Bettina Mattei, Tina C. Mcintosh, Michael P. Mcleod, Duncan Mcpherson, Gennady Merkulov, Natalia V. Milshina, Clark Mobarry, Joe Morris, Ali Moshrefi, Stephen M. Mount, Mee Moy, Brian Murphy, Lee Murphy, Donna M. Muzny, David L. Nelson, David R. Nelson, Keith A. Nelson, Katherine Nixon, Deborah R. Nusskern, Joanne M. Pacleb, Michael Palazzolo, Gjange S. Pittman, Sue Pan, John Pollard, Vinita Puri, Martin G. Reese, Knut Reinert, Karin Remington, Robert D. Saunders, Frederick Scheeler, Hua Shen, Bixiang C. Shue, Inga Siden-Kiamos, Michael Simpson, Marian P. Skupski, Tom Smith, Eugene Spier, Allan C. Spradling, Mark Stapleton, Renee Strong, Eric Sun, Robert Svirskas, Cyndee Tector, Russell Turner, Eli Venter, Aihui H. Wang, Xin Wang, Zhen-Yuan Wang, David A.

Wassarman, George M. Weinstock, Jean Weissenbach, Sherita M. Williams, Trevor Woodage, Kim C. Worley, David Wu, Song Yang, Q. Alison Yao, Jane Ye, Ru-Fang Yeh, Jayshree S. Zaveri, Ming Zhan, Guangren Zhang, Qi Zhao, Liansheng Zheng, Xiangqun H. Zheng, Fei N. Zhong, Wenyan Zhong, Xiaojun Zhou, Shiaoping Zhu, Xiaohong Zhu, Hamilton O. Smith, Richard A. Gibbs, Eugene W. Myers, Gerald M. Rubin, and J. Craig Venter. The genome sequence of drosophila melanogaster. *Science*, 287(5461):2185–2195, March 2000. doi: 10.1126/science.287.5461.2185. URL `http://dx.doi.org/10.1126/science.287.5461.2185`.

K. A. Al-Kofahi, S. Lasek, D. H. Szarowski, C. J. Pace, G. Nagy, J. N. Turner, and B. Roysam. Rapid automated three-dimensional tracing of neurons from confocal image stacks. *IEEE Trans Inf Technol Biomed*, 6:171–187, Jun 2002.

Y. Al-Kofahi, N. Dowell-Mesfin, C. Pace, W. Shain, J. N. Turner, and B. Roysam. Improved detection of branching points in algorithms for automated neuron tracing from 3D confocal images. *Cytometry A*, 73:36–43, Jan 2008.

J. D. Armstrong and J. I. van Hemert. Towards a virtual fly brain. *Philosophical Transactions A*, 367:2387–2397, Jun 2009.

J D Armstrong, M J Texada, R Munjaal, D A Baker, and K M Beckingham. Gravitaxis in Drosophila melanogaster: a forward genetic screen. *Genes, Brain and Behavior*, 5(3):222–239, 2006.

Michael Ashburner. *Won for All: How the Drosophila Genome Was Sequenced*. Cold Spring Harbor Laboratory Press, March 2006. ISBN 0879698020.

K Basler and G Struhl. Compartment boundaries and the control of Drosophila limb pattern by hedgehog protein. *Nature*, 368(6468):208–14, 1994.

M.F. Bear, B.W. Connors, and M.A. Paradiso. *Neuroscience: Exploring the brain*. Lippincott Williams & Wilkins, 2006.

B M Bolstad, R A Irizarry, M Astrand, and T P Speed. A comparison of normalization methods for high density oligonucleotide array data based on variance and bias. *Bioinformatics*, 19(2):185–93, 2003.

F. L. Bookstein. Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11(6):567–585, 1989. ISSN 0162-8828. doi: http://dx.doi.org/10.1109/34.24792.

A. H. Brand and N. Perrimon. Targeted gene expression as a means of altering cell fates and generating dominant phenotypes. *Development*, 118:401–415, Jun 1993.

R. Brandt, T. Rohlfing, J. Rybak, S. Krofczik, A. Maye, M. Westerhoff, H. C. Hege, and R. Menzel. Three-dimensional average-shape atlas of the honeybee brain and its applications. *J. Comp. Neurol.*, 492:1–19, Nov 2005.

K. L. Briggman and W. Denk. Towards neural circuit reconstruction with volume electron microscopy techniques. *Curr. Opin. Neurobiol.*, 16:562–570, Oct 2006.

Matthew Brown, Richard Szeliski, and Simon Winder. Multi-Image Matching using Multi-Scale Oriented Patches. *Microsoft Research Technical Report*, December 2004. URL `http://research.microsoft.com/apps/pubs/default.aspx?id=70120`.

J.J. Capowski. *Computer techniques in neuroanatomy.* Plenum Press New York, NY, USA, 1989.

A. Carhan, F. Allen, J. D. Armstrong, S. F. Goodwin, and K. M. C. O'Dell. Female receptivity phenotype of icebox mutants caused by a mutation in the L1-type cell adhesion molecule neuroglian. *Genes, Brain and Behavior*, 4(8):449–465, 2005.

Vicent Caselles, Ron Kimmel, and Guillermo Sapiro. Geodesic active contours. *International Journal of Computer Vision*, 22(1):61–79, 1997.

G. E. Christensen, R. D. Rabbitt, and M. I. Miller. 3D brain mapping using a deformable neuroanatomy. *Phys Med Biol*, 39:609–618, Mar 1994.

I. E. Clark, L. Y. Jan, and Y. N. Jan. Reciprocal localization of Nod and kinesin fusion proteins indicates microtubule polarity in the Drosophila oocyte, epithelium, neuron and muscle. *Development*, 124:461–470, Jan 1997.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Cliff Stein. *Introduction to Algorithms.* McGraw-Hill, 2nd edition, 2001.

S. V. Costes, D. Daelemans, E. H. Cho, Z. Dobbin, G. Pavlakis, and S. Lockett. Automatic and quantitative measurement of protein-protein colocalization in live cells. *Biophys. J.*, 86:3993–4003, Jun 2004.

Douglas Cowan. Tools for remote analysis and handling of medical images. Master's thesis, University of Edinburgh, 2003.

M. H. Davis, A. Khotanzad, D. P. Flamig, and S. E. Harms. A physics-based coordinate transformation for 3-D image matching. *IEEE Trans Med Imaging*, 16:317–328, Jun 1997.

J. S. de Belle and M. Heisenberg. Associative odor learning in Drosophila abolished by chemical ablation of mushroom bodies. *Science*, 4(5147):692–695, 1994.

Paul E. Debevec and Jitendra Malik. Recovering high dynamic range radiance maps from photographs. *Computer Graphics*, 31(Annual Conference Series):369–378, 1997. URL `http://citeseer.ist.psu.edu/debevec97recovering.html`.

J. Dubnau and T. Tully. Gene discovery in Drosophila: new insights for learning and memory. *Annu. Rev. Neurosci.*, 21:407–444, 1998.

J. B. Duffy. GAL4 system in Drosophila: a fly geneticist's Swiss army knife. *Genesis*, 34:1–15, 2002.

R. D. Emes, A. J. Pocklington, C. N. Anderson, A. Bayes, M. O. Collins, C. A. Vickers, M. D. Croning, B. R. Malik, J. S. Choudhary, J. D. Armstrong, and S. G. Grant. Evolutionary expansion and anatomical specialization of synapse proteome complexity. *Nat. Neurosci.*, 11:799–806, Jul 2008.

J.F. Evers, S. Schmitt, M. Sibila, and C. Duch. Progress in functional neuroanatomy: precise automatic geometric reconstruction of neuronal morphology from confocal image stacks. *J. Neurophysiol.*, 93:2331–2342, Apr 2005.

Mike Fornefett, Karl Rohr, H. Siegfried Stiehl, and Arbeitsbereich Kognitive Systeme. Radial basis functions with compact support for elastic registration of medical images. *IVC*, 19:1–2, 2001.

W. Goll. Strukturuntersuchungen am Gehirn von Formica. *Zoomorphology*, 59(2):143–210, 1967.

Ralph J. Greenspan. *Fly Pushing*. Cold Spring Harbor Laboratory Press, third edition, 1997. ISBN 0879694920.

U. Hanesch, K.-F. Fischbach, and M. Heisenberg. Neuronal architecture of the central complex in drosophila melanogaster. *Cell Tissue Research*, 257:343–366, 1989.

P.E. Hart, N.J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

S. Heinze and U. Homberg. Neuroarchitecture of the central complex of the desert locust: Intrinsic and columnar neurons. *J. Comp. Neurol.*, 511:454–478, Dec 2008.

M. Heisenberg, A. Borst, S. Wagner, and D. Byers. Drosophila mushroom body mutants are deficient in olfactory learning. *Journal of Neurogenetics*, 2(1):1–30, 1985.

Martin Heisenberg. Mushroom Body Memoir: From Maps To Models. *Nature Reviews Neuroscience*, 4(4):266–275, April 2003.

M. Holden. A review of geometric transformations for nonrigid body registration. *IEEE Trans Med Imaging*, 27:111–128, Jan 2008.

Berthold K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4(4):629–642, 1987.

P.O. Hoyer. Non-negative matrix factorization with sparseness constraints. *The Journal of Machine Learning Research*, 5:1457–1469, 2004.

H. Husi, M. A. Ward, J. S. Choudhary, W. P. Blackstock, and S. G. Grant. Proteomic analysis of NMDA receptor-adhesion protein signaling complexes. *Nat. Neurosci.*, 3:661–669, Jul 2000.

A. Hyvärinen. Fast and robust fixed-point algorithms for independent component analysis. *IEEE Transactions on Neural Networks*, 10(3):626–634, 1999.

A. Hyvärinen and E. Oja. Independent component analysis: algorithms and applications. *Neural networks*, 13(4-5):411–430, 2000.

M. Ilius, R. Wolf, and M. Heisenberg. The central complex of Drosophila melanogaster is involved in flight control: studies on mutants and mosaics of the gene ellipsoid body open. *J. Neurogenet.*, 9 (3):189–206, 1994.

G. Isabel, A. Pascual, and T. Preat. Exclusive consolidated memory phases in Drosophila. *Science*, 304:1024–1027, May 2004.

K. Ito, R. Okada, N.K. Tanaka, and T. Awasaki. Cautionary observations on preparing and interpreting brain images using molecular biology-based staining techniques. *Microsc. Res. Tech.*, 62:170–186, Oct 2003. URL `http://doi.wiley.com/10.1002/jemt.10369`.

J. Sambrook and E. F. Fritsch and T. Maniatis. *Molecular Cloning: A Laboratory Manual*. Cold Spring Harbor Laboratory Press, 2nd edition, 1987.

G. S. Jefferis, C. J. Potter, A. M. Chan, E. C. Marin, T. Rohlfing, C. R. Maurer, and L. Luo. Comprehensive maps of Drosophila higher olfactory centers: spatially segregated fruit and pheromone representation. *Cell*, 128:1187–1203, Mar 2007.

A. Jeibmann and W. Paulus. Drosophila melanogaster as a Model Organism of Brain Diseases. *Int J Mol Sci*, 10:407–440, Feb 2009.

A. Jenett, J. E. Schindelin, and M. Heisenberg. The Virtual Insect Brain protocol: creating and comparing standardized neuroanatomy. *BMC Bioinformatics*, 7:544, 2006.

Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

A. C. Keene and S. Waddell. Drosophila olfactory memory: single genes to complex neural circuits. *Nat. Rev. Neurosci.*, 8:341–354, May 2007.

T. Kitamoto. Conditional modification of behavior in Drosophila by targeted expression of a temperature-sensitive shibire allele in defined neurons. *J. Neurobiol.*, 47:81–92, May 2001.

Seymour Knowles-Barley. BrainTrap: Drosophila Melanogaster Brain Protein Database. Master's thesis, University of Edinburgh, 2007. http://cmor.net/masters.pdf.

Jan Kohlrausch, Karl Rohr, and H. Siegfried Stiehl. A new class of elastic body splines for nonrigid registration of medical images. *J. Math. Imaging Vis.*, 23(3):253–280, 2005. ISSN 0924-9907. doi: http://dx.doi.org/10.1007/s10851-005-0483-7.

A. E. Kurylas, T. Rohlfing, S. Krofczik, A. Jenett, and U. Homberg. Standardized atlas of the brain of the desert locust, Schistocerca gregaria. *Cell Tissue Res.*, 333:125–145, Jul 2008.

Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In *NIPS*, pages 556–562, 2000. URL `http://citeseer.ist.psu.edu/lee01algorithms.html`.

D.D. Lee and H.S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

T. Lee and L. Luo. Mosaic analysis with a repressible cell marker (MARCM) for Drosophila neural development. *Trends Neurosci.*, 24:251–254, May 2001.

Ta-Chih Lee, Rangasami L. Kashyap, and Chong-Nam Chu. Building skeleton models via 3-d medial surface/axis thinning algorithms. *CVGIP: Graph. Models Image Process.*, 56(6):462–478, 1994. ISSN 1049-9652. doi: http://dx.doi.org/10.1006/cgip.1994.1042.

W. Li, Y. Pan, Z. Wang, H. Gong, Z. Gong, and L. Liu. Morphological characterization of single fan-shaped body neurons in Drosophila melanogaster. *Cell Tissue Res.*, 336:509–519, Jun 2009.

J. W. Lichtman, J. Livet, and J. R. Sanes. A technicolour approach to the connectome. *Nat. Rev. Neurosci.*, 9:417–422, Jun 2008.

S. Q. Lima and G. Miesenböck. Remote control of behavior through genetically targeted photostimulation of neurons. *Cell*, 121:141–152, Apr 2005.

Tony Lindeberg. Scale-space theory: A basic tool for analysing structures at different scales. *Journal of Applied Statistics*, 21(2):225–270, 1994.

Gang Liu, Holger Seiler, Ai Wen, Troy Zars, Kei Ito, Reinhard Wolf, Martin Heisenberg, and Li Liu. Distinct memory traces for two visual features in the Drosophila brain. *Nature*, 439(7076):551–556, Feb 2006.

J. Livet, T. A. Weissman, H. Kang, R. W. Draft, J. Lu, R. A. Bennis, J. R. Sanes, and J. W. Lichtman. Transgenic strategies for combinatorial expression of fluorescent proteins in the nervous system. *Nature*, 450:56–62, Nov 2007.

Mark H. Longair. Immunohistochemical Assays for the Brain of Drosophila Melanogaster. Master's thesis, University of Edinburgh, October 2004.

David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

J. R. Martin, T. Raabe, and M. Heisenberg. Central complex substructures are required for the maintenance of locomotor activity in Drosophila melanogaster. *J. Comp. Physiol. A*, 185:277–288, Sep 1999.

Leeanne McGurk, Harris Morrison, Liam P. Keegan, James Sharpe, and Mary A. O'Connell. Three-dimensional imaging of drosophila melanogaster. *PLoS ONE*, 2(9):e834, 2007. doi: 10.1371/journal.pone.0000834. URL `http://dx.plos.org/10.1371/journal.pone.0000834`.

E. Meijering, M. Jacob, J.-C. F. Sarria, P. Steiner, H. Hirling, and M. Unser. Design and validation of a tool for neurite tracing and analysis in fluorescence microscopy images. *Cytometry*, 58A(2): 167–176, April 2004. URL `http://www.imagescience.org/meijering/software/neuronj/`.

Jorge Nocedal and Stephen J Wright. *Numerical Optimization*. Springer, 2nd edition, 2006.

OED. *The Oxford English Dictionary*. Oxford University Press, 2nd edition, 1998. URL `http://dictionary.oed.com/cgi/entry/50201236`.

Stephen M. Pizer, Kaleem Siddiqi, Gabor Székely, James N. Damon, and Steven W. Zucker. Multiscale medial loci and their properties. *Int. J. Comput. Vision*, 55(2-3):155–179, 2003. URL `http://portal.acm.org/citation.cfm?id=945893`.

A. J. Pocklington, J. D. Armstrong, and S. G. Grant. Organization of brain complexity–synapse proteome form and function. *Brief Funct Genomic Proteomic*, 5:66–73, Mar 2006.

Stephan Preibisch. Fast Stitching of Huge 3D Biological Datasets, 2008.

Karlheinz Rein, Malte Zockler, Michael T Mader, Cornelia Grubel, and Martin Heisenberg. The Drosophila standard brain. *Curr Biol*, 12(3):227–231, Feb 2002.

A. Reiner, D. J. Perkel, L. L. Bruce, A. B. Butler, A. Csillag, W. Kuenzel, L. Medina, G. Paxinos, T. Shimizu, G. Striedter, M. Wild, G. F. Ball, S. Durand, O. Güntürkün, D. W. Lee, C. V. Mello, A. Powers, S. A. White, G. Hough, L. Kubikova, T. V. Smulders, K. Wada, J. Dugas-Ford, S. Husband, K. Yamamoto, J. Yu, C. Siang, E. D. Jarvis, and O. Gütürkün. Revised nomenclature for avian telencephalon and some related brainstem nuclei. *J. Comp. Neurol.*, 473:377–414, May 2004.

Susan C.P. Renn, J. Douglas Armstrong, Mingyao Yang, Zongsheng Wang, Xin An, Kim Kaiser, and Paul H. Taghert. Genetic Analysis of the Drosophila Ellipsoid Body Neuropil: Organization and Development of the Central Complex. *Journal of Neurobiology*, 41(2):189–207, 1999.

T. Rohlfing and Jr. C. R. Maurer. Nonrigid image registration in shared-memory multiprocessor environments with application to brains, breasts, and bees. *IEEE Transactions on Information Technology in Biomedicine*, 7(1):16–25, 2003.

T. Rohlfing and C. R. Maurer. Shape-based averaging. *IEEE Trans Image Process*, 16:153–161, Jan 2007.

Torsten Rohlfing, Robert Br, Calvin R. Maurer, and Olf Menzel. Bee brains, b-splines and computational democracy: Generating an average shape atlas. In *in IEEE Workshop on Mathematical Methods in Biomedical Image Analysis*, pages 187–194, 2001.

D. Rueckert, L. I. Sonoda, C. Hayes, D. L. Hill, M. O. Leach, and D. J. Hawkes. Nonrigid registration using free-form deformations: application to breast MR images. *IEEE Trans Med Imaging*, 18: 712–721, Aug 1999.

T. Sakai and T. Kitamoto. Differential roles of two major brain structures, mushroom bodies and central complex, for Drosophila male courtship behavior. *J. Neurobiol.*, 66:821–834, Jul 2006.

N. Sánchez-Soriano, W. Bottenberg, A. Fiala, U. Haessler, A. Kerassoviti, E. Knust, R. Löhr, and A. Prokop. Are dendrites in Drosophila homologous to vertebrate dendrites? *Dev. Biol.*, 288: 126–138, Dec 2005.

Y. Sato, S. Nakajima, N. Shiraga, H. Atsumi, S. Yoshida, T. Koller, G. Gerig, and R. Kikinis. Three-dimensional multi-scale line filter for segmentation and visualization of curvilinear structures in medical images. *Med Image Anal*, 2:143–168, Jun 1998.

S. Schmitt, J.F. Evers, C. Duch, M. Scholz, and K. Obermayer. New methods for the computer-assisted 3-D reconstruction of neurons from confocal image stacks. *Neuroimage*, 23:1283–1298, Dec 2004.

H. Scholz, J. Ramond, C. M. Singh, and U. Heberlein. Functional ethanol tolerance in Drosophila. *Neuron*, 28:261–271, Oct 2000.

J. A. Sethian. *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge University Press, June 1999.

D. St Johnston. The art and design of genetic screens: Drosophila melanogaster. *Nat. Rev. Genet.*, 3:176–188, Mar 2002.

N. J. Strausfeld. A brain region in insects that supervises walking. *Prog. Brain Res.*, 123:273–284, 1999.

Nicholas J. Strausfeld. *Atlas of an Insect Brain* . Springer-Verlag, 1976.

R. Strauss. The central complex and the genetic dissection of locomotor behaviour. *Current Opinion in Neurobiology*, 12(6):633–638, December 2002.

R. Strauss and M. Heisenberg. A higher control center of locomotor behavior in the Drosophila brain. *J. Neurosci.*, 13:1852–1861, May 1993.

R. Strauss, U. Hanesch, M. Kinkelin, R. Wolf, and M. Heisenberg. No-bridge of Drosophila melanogaster: portrait of a structural brain mutant of the central complex. *J. Neurogenet.*, 8: 125–155, Sep 1992.

J. P. Thirion. Image matching as a diffusion process: an analogy with maxwell's demons. *Medical Image Analysis*, pages 243–260, September 1998. ISSN 1361-8415. doi: http://dx.doi.org/10.1016/S1361-8415(98)80022-4. URL `http://dx.doi.org/10.1016/S1361-8415(98)80022-4`.

Arthur W Toga. *Brain Warping*. Academic Press, 1999.

H. Vitzthum, M. Muller, and U. Homberg. Neurons of the central complex of the locust Schistocerca gregaria are sensitive to polarized light. *J. Neurosci.*, 22:1114–1125, Feb 2002.

L. B. Vosshall. Into the mind of a fly. *Nature*, 450:193–197, Nov 2007.

D. A. Wagh, T. M. Rasse, E. Asan, A. Hofbauer, I. Schwenkert, H. Dürrbeck, S. Buchner, M. C. Dabauvalle, M. Schmidt, G. Qin, C. Wichmann, R. Kittel, S. J. Sigrist, and E. Buchner. Bruchpilot, a protein with homology to ELKS/CAST, is required for structural integrity and function of synaptic active zones in Drosophila. *Neuron*, 49:833–844, Mar 2006.

J. Wang, X. Ma, J. S. Yang, X. Zheng, C. T. Zugates, C. H. Lee, and T. Lee. Transmembrane/juxtamembrane domain-dependent Dscam distribution and function during mushroom body neuronal morphogenesis. *Neuron*, 43:663–672, Sep 2004.

Z. Wang, Y. Pan, W. Li, H. Jiang, L. Chatzimanolis, J. Chang, Z. Gong, and L. Liu. Visual pattern memory requires foraging function in the central complex of Drosophila. *Learn. Mem.*, 15:133–142, Mar 2008.

Jonathan Weiner. *Time, Love, Memory.* Faber and Faber, October 2000.

J.L.D. Williams. Anatomical studies of the insect central nervous system: a ground-plan of the mid-brain and an introduction to the central complex in the locust, Schistocerca gregaria (Orthoptera). *Journal of Zoology*, pages 67–86, 1975.

O. Wink, W.J. Niessen, and M. A. Viergever. Minimum cost path determination using a simple heuristic function. In *ICPR '00: Proceedings of the International Conference on Pattern Recognition*, pages 1010–1013, Washington, DC, USA, 2000. IEEE Computer Society.

Reinhard Wolf, Tobias Wittig, Li Liu, Gerold Wustmann, Dirk Eyding, and Martin Heisenberg. Drosophila Mushroom Bodies Are Dispensable for Visual, Tactile, and Motor Learning. *Learning and Memory*, 5(1):166–178, May/June 1998.

Allan M Wong, Jing W Wang, and Richard Axel. Spatial representation of the glomerular map in the Drosophila protocerebrum. *Cell*, 109(2):229–41, 2002.

M. Y. Yang, J. D. Armstrong, I. Vilinsky, N. J. Strausfeld, and K. Kaiser. Subdivision of the Drosophila mushroom bodies by enhancer-trap expression patterns. *Neuron*, 15:45–54, Jul 1995.

J. Young and J. D. Armstrong. The structure of the adult central complex in drosophila: Organisation of distinct neuronal subsets. 2009, in preparation.

T. Zars, R. Wolf, R. Davis, and M. Heisenberg. Tissue-specific expression of a type I adenylyl cyclase rescues the rutabaga mutant memory defect: in search of the engram. *Learn. Mem.*, 7:18–31, Jan 2000.